Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Žan Hvala

Izdelava spletne aplikacije za iskanje izgubljenih predmetov

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: viš. pred. dr. Igor Rožanc

Ljubljana, 2017

COPYRIGHT. Rezultati diplomske naloge so intelektualna lastnina avtorja in Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavo in koriščenje rezultatov diplomske naloge je potrebno pisno privoljenje avtorja, Fakultete za računalništvo in informatiko ter mentorja.

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomski nalogi predstavite sistematičen razvoj prototipa spletne aplikacije za iskanje izgubljenih predmetov. Cilj naj bo izdelati krovno spletno stran za iskanje izgubljenih predmetov v Sloveniji, zato preučite tudi specifične dodatne zahteve kot so recimo možnosti za uvoz podatkov o izgubljenih predmetih iz že obstoječih spletnih aplikacij. V nalogi najprej preučite in predstavite obstoječe rešitve, določite zahteve in izberite najprimernejšo tehnologijo za razvoj spletne aplikacije. Aplikacijo sistematično načrtujte, izdelajte delujoč prototip in ga testirajte. Aplikacijo poskusno namestite ter analizirajte njeno uporabnost.

Za pomoč pri izdelavi diplomskega dela se zahvaljujem mentorju, viš. pred. dr. Igorju Rožancu. Posebna zahvala gre družini in dekletu Maši, ki so me podpirali celoten čas študija.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
API	Application Programming Interface	aplikacijski programski vmesnik
CSS	Cascading Style Sheets	predloge, ki določajo izgled spletnih strani
EJB	Enterprise JavaBeans	poslovna Java zrna
ER diagram	Entity Relationship Diagram	entitetno relacijski diagram
HTML	Hyper Text Markup Language	jezik za označevanje nadbesedila
HTTP	Hypertext Transfer Protocol	protokol za prenos hiperteksta
Java EE	Java Enterprise Edition	poslovna različica Jave
JAAS	Java Authentication and Authorization Service	Java servis avtentikacije in avtorizacije
JSON	JavaScript Object Notation	format označevanja JavaScript objektov
JVM	Java virtual machine	Java virtualni stroj
MVC	Model–view–controller	model-pogled-krmilnik
SQL	Structured Query Language	strukturiran poizvedovalni jezik
URL	Uniform Resource Locator	enolični krajevnik vira
XML	Extensible Markup Language	razširljiv označevalni jezik

Povzetek

Namen diplomske naloge je prikaz poteka izdelave prototipa spletne aplikacije za iskanje izgubljenih in najdenih predmetov. Glavne naloge spletne aplikacije so hitro in enostavno iskanje predmetov in dodajanje novih, najdenih ali izgubljenih predmetov. V začetni fazi so bile pregledane že obstoječe spletne aplikacije za iskanje najdenih in izgubljenih predmetov. Zbrane so bile funkcionalne in nefunkcionalne zahteve, s pomočjo katerih se je lahko pričel sam razvoj. Spletna aplikacija je razvita v programskem jeziku Java EE. Za hranjenje podatkov uporablja podatkovno bazo MySQL, aplikacijski strežnik WildFly pa omogoča nemoteno izvajanje same aplikacije. Izdelana spletna aplikacija omogoča prijavo in registracijo uporabnikov, iskanje in dodajanje izgubljenih in najdenih predmetov, administracijo uporabnikov, ogled revizijske sledi posameznega predmeta, uvoz podatkov o predmetih iz drugih spletnih strani ter možnost obnovitve predmetov v primeru izgube podatkov ali napake v aplikaciji. Narejena je bila analiza uporabnosti spletne aplikacije. Ta preuči in utemelji njeno dejansko vrednost.

Ključne besede: spletna aplikacija, Java EE, MySQL, WildFly, iskanje izgubljenih predmetov.

Abstract

Title: The development of the web application for searching of lost items

Author: Žan Hvala

The purpose of the thesis is to present the process of making prototype of web application for finding lost and found items. The main tasks of web application are enabling quick and easy search of items and adding new lost or found items to the list. In the initial phase functional and nonfunctional requirements of application were collected. With this requirements in hand the process of implementation is able to start. Web application is based on Java EE programming language. MySQL database is used for storing data. WildFly application server ensures web application to run smoothly. In the final form application functionalities are: log in, log out, register account, search for lost or found items, administrate registered users, view audit trail of each item, import items data from other web pages and recover items data in case of application failure or data loss. An analysis of application usability was made, which examined and justified its actual value.

Keywords: web application, Java EE, MySQL, WildFly, finding lost items.

Poglavje 1

Uvod

Danes so najrazličnejše spletne aplikacije zelo razširjene in dostopne kadarkoli in kjerkoli. Vseeno pa v času izdelave diplomske naloge ni bilo mogoče najti spletne aplikacije v slovenskem jeziku, ki bi bila namenjena izključno iskanju izgubljenih in najdenih predmetov. Najdemo lahko le spletne aplikacije, ki ponujajo iskanje predmetov kot dodatek k osnovni funkcionalnosti. Iz tega lahko sklepamo, da iskanje predmetov ni ključnega pomena za omenjene aplikacije, kar se odraža tudi pri sami uporabi iskanja. To ni tako dodelano in intuitivno kot bi si želeli. Zato smo izdelali aplikacijo za enostavno in hitro iskanje predmetov, ki ima ambicijo postati krovna spletna aplikacija v ta namen.

Cilj je izdelati prototip spletne aplikacije, katere glavna naloga je iskanje, dodajanje in urejanje izgubljenih in najdenih predmetov, ki so dodani preko same aplikacije ali pridobljeni iz drugih spletnih strani. Uporabniški vmesnik mora biti kar se da odziven in enostaven za uporabo. To omogoča potencialnim uporabnikom, da lahko z le nekaj kliki dostopajo do vseh glavnih funkcionalnosti aplikacije in tako ne izgubljajo dragocenega časa. Spletna aplikacija mora omogočati uvoz in izvoz predmetov. Za boljšo preglednost nad predmeti se mora le te po določenem času odstraniti, saj niso več aktualni. Kljub izbrisu se podatki o predmetih ne smejo izgubiti, zato je potrebno arhiviranje. Za nadzor nad predmeti mora za vsak predmet obstajati revi-

zijska sled.

Izdelave spletne aplikacije smo se lotili s pomočjo programskega jezika Java EE. Omogoča nam izdelavo varne, hitre in zanesljive aplikacije. Uporabljene so tudi različne knjižnice, ki so namenjene hitrejši in lažji izdelavi uprabniškega vmesnika, manipuliranju z nizi in slikami, pošiljanju elektronskih sporočil in lažjemu delu z JSON formatom [11]. Podatki so shranjeni v MySQL [19] podatkovni bazi, ki je odprtokodna implementacija relacijske podatkovne baze. Upravljanje nad podatki v bazi izvajamo s pomočjo jezika SQL. Spletna aplikacija teče na aplikacijskem strežniku WildFly [7], ki podpira tehnologije specifikacije Java EE 7 in omogoča kontrolo dostopa na podlagi vlog.

Po izdelavi smo analizo uporabnosti in tako preverili, če naša rešitev ustreza zahtevam. V sklopu anlize smo preučili, ali so izpolnjene vse funkcionalne in nefunkcionalne zahteve, preverili odzivnost aplikacije ter testirali njeno uporabnost pri potencialnih uporabnikih.

S ciljem da aplikacija postane osrednje mesto za iskanje predmetov, želimo našo spletno aplikacijo ponuditi v uporabo širši množici. V iskalnikih se mora aplikacija uvrstiti na čim višje mesto, saj bi s tem gradili na prepoznavnosti. Nove uporabnike bi pridobivali tudi z oglaševanjem in sprotnim dodanjanjem novih funkcionalnosti.

V nadaljevanju diplomske naloge je predstavljen razvoj naše aplikacije in z njo povezanih tehnologij. V drugem poglavju so opisane že obstoječe rešitve, ki se navezujejo na temo iskanja izgubljenih in najdenih predmetov. V naslednjem poglavju so podrobno opisane tehnologije uporabljene pri izdelavi. Sledi poglavje, ki je namenjeno predstavitvi funkcionalnih in nefunkcionalnih zahtev ter načrta aplikacije. Opis poteka implementacije je predstavljen v petem poglavju, kateremu sledita še analiza in zaključek.

Poglavje 2

Obstoječe aplikacije za iskanje predmetov

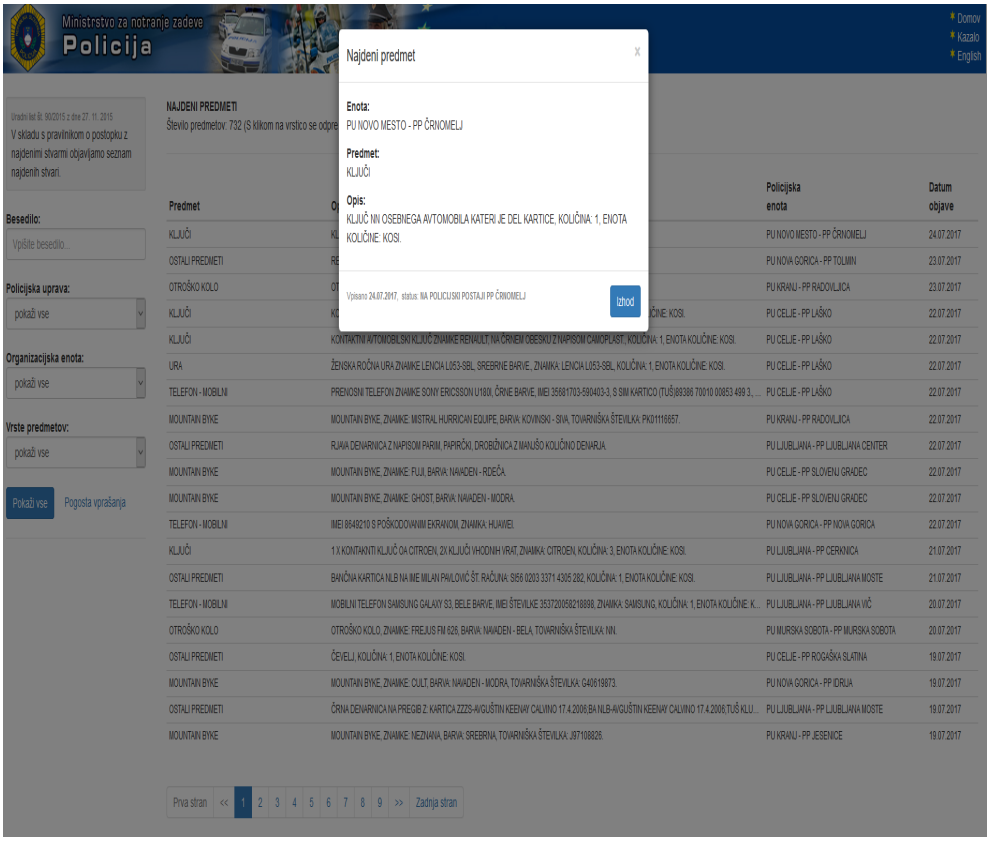
Pred začetkom izdelave diplomskega dela smo preučili spletne aplikacije, katere omogočajo iskanje izgubljenih predmetov, vendar to ni nujno njihova glavna funkcionalnost. Med seboj smo primerjali funkcionalnosti, ki jih posamezna aplikacija omogoča v okviru iskanja predmetov. Nekatere aplikacije ne omogočajo iskanja po seznamu predmetov, dodajanja novih predmetov, spreminjanja podatkov predmeta in ostalih pomembnih funkcionalnosti, ki bi jih od takih aplikacij pričakovali.

2.1 Spletne aplikacije z možnostjo iskanja predmetov

Na spletu smo poiskali aplikacije z možnostjo iskanja predmetov in našli sledeče:

- **Spletna stran policije** [41] (slika 2.1) omogoča pregled najdenih predmetov z možnostjo filtriranja po organizacijski enoti, policijski upravi ali vrsti predmeta ter iskanje predmetov glede na vpisano besedilo. Vrstica v seznamu predstavlja posamezen predmet, katerega lastnosti so

oznaka, opis, policijska enota, kjer se predmet nahaja, in datum objave. S klikom na vrstico se nam prikaže pojavno okno, v katerem so podatki o predmetu prikazani v celoti, saj daljša besedila pri opisu predmeta presegajo dolžino stolpca, ki je namenjen prikazu besedila opisa.



Slika 2.1: Spletna stran policije.

Vsi podatki o najdenih predmetih so zapisani tudi v JSON formatu.[1] Programska koda prikazuje najdene predmete s strani policije predstavljene v prej omenjenem formatu.

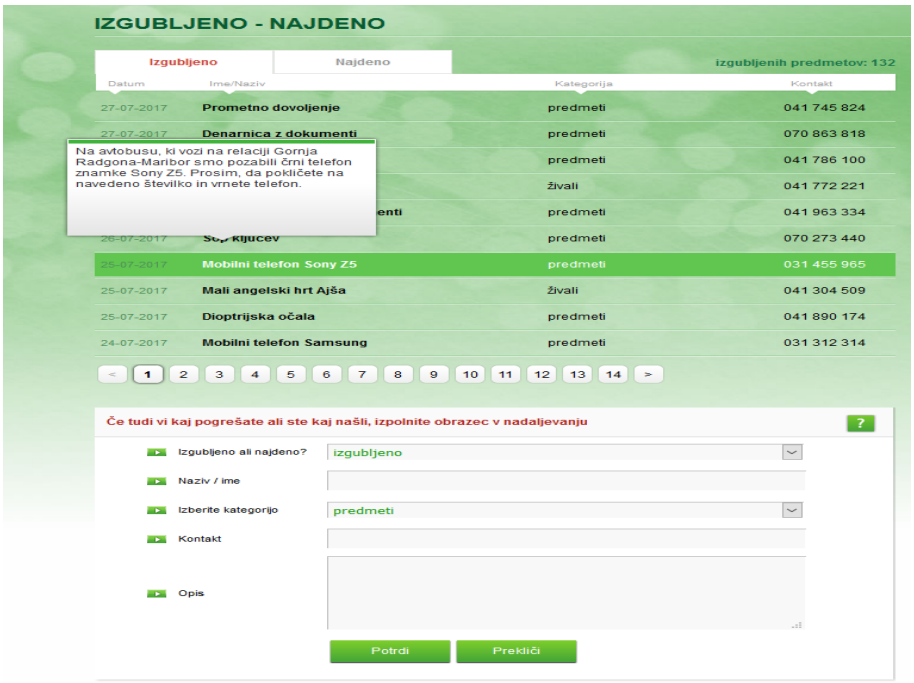
[1]Do seznama predmetov predmetov lahko dostopamo preko povezave `http://www.policija.si/apps/predmeti/rest.php/najdeni`.

Najdeni predmeti predstavljeni v JSON formatu.


```
[
  {
    "ST": "5026549",
    "PU": "PU KOPER",
    "OE": "PP PIRAN",
    "PREDMET": "NAVADNO KOLO",
    "OPIS": "NAVADNO KOLO, ZNAMKE: CYCO LISE, BARVA:
       NAVADEN - ČRNA, TOVARNIŠKA ŠTEVILKA:
       MAQ1109270041393.",
    "DATUM_VPIS": "2017-08-16",
    "STATUS": "NA POLICIJSKI POSTAJI PP PIRAN"
  },
  {
    "ST": "5026551",
    "PU": "PU KOPER",
    "OE": "PP PIRAN",
    "PREDMET": "MOUNTAIN BYKE",
    "OPIS": "MOUNTAIN BYKE, ZNAMKE: IDEAL, BARVA: NAVADEN
       - BELA, TOVARNIŠKA ŠTEVILKA: WAZ7.",
    "DATUM_VPIS": "2017-08-16",
    "STATUS": "NA POLICIJSKI POSTAJI PP PIRAN"
  },
...
```


- **Spletna stran Radio City [42]** (slika 2.2) omogoča pregled najdenih in izgubljenih predmetov. Prikaz predmetov je narejen na podoben način kot na spletni strani policije. Ob premiku miškinega kazalca na vrstico, ki predstavlja predmet, se pojavi celoten opis predmeta, vendar nanj ne moremo klikniti. Spletna stran ponuja tudi možnost dodajanja izgubljenega ali najdenega predmeta na seznam. To naredimo tako, da izpolnemo obrazec, v katerem izberemo, ali je predmet izgubljen ali najden, vpišemo naziv predmeta, izberemo kategorijo ter podamo naš kontakt in opis predmeta.

  Podatki o najdenih in izgubljenih predmetih so predstavljeni izključno



Slika 2.2: Spletna stran Radio City.

v HTML obliki. Primer prikaza podatkov predmeta na spletni strani Radio City ponazarja spodnja programska koda.

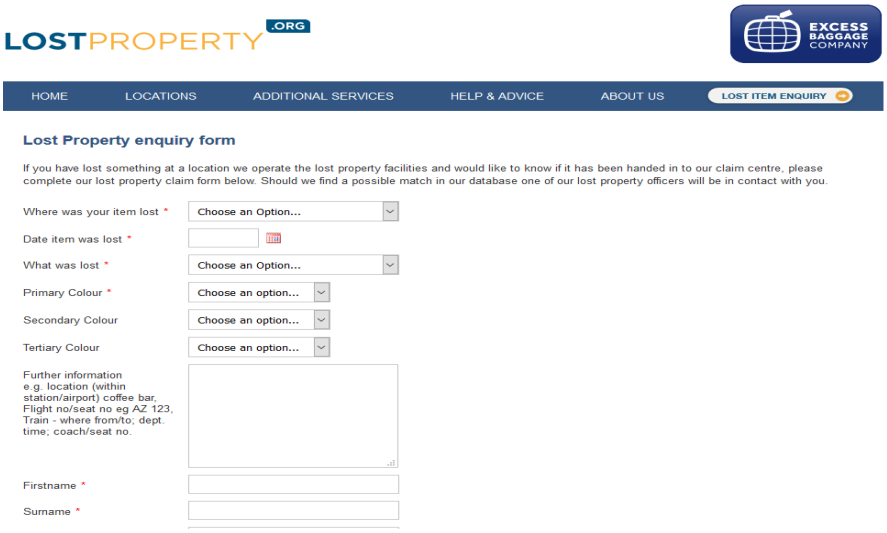
Predmeti predstavljeni v HTML obliki.

```
<div id="izgubljeno1" class="izgubljeno-list">
  <div class="izgubljeno-datum">15-08-2017</div>
  <div id="item1" class="izgubljeno-naziv"><strong>Ženska
     očala z dioptrijo</strong></div>
  <div class="izgubljeno-kategorija">predmeti</div>
  <div class="izgubljeno-kontakt">031 806 152</div>
  <script>
    $("#item1").qtip({ hide: { effect: 'slide', when:
       'mouseout', fixed: true }, show: { effect: 'slide'
       }, content: '<div class="tooltiptop"> Izgubila sem ž
       enska očala znamke Ray ban z dioptrijo.</div>',
       position: { corner: { target: 'topMiddle', tooltip:
       'bottomMiddle' }, adjust: { x: -130, y: -10}},
```

```
        style: { width: { max: 275 }, background:
        'transparent', padding: '1px', border: { width: 0,
        radius: 0, color: '#4c4c4c' }} });
  </script>
  <br style="clear:both" />
</div>
```

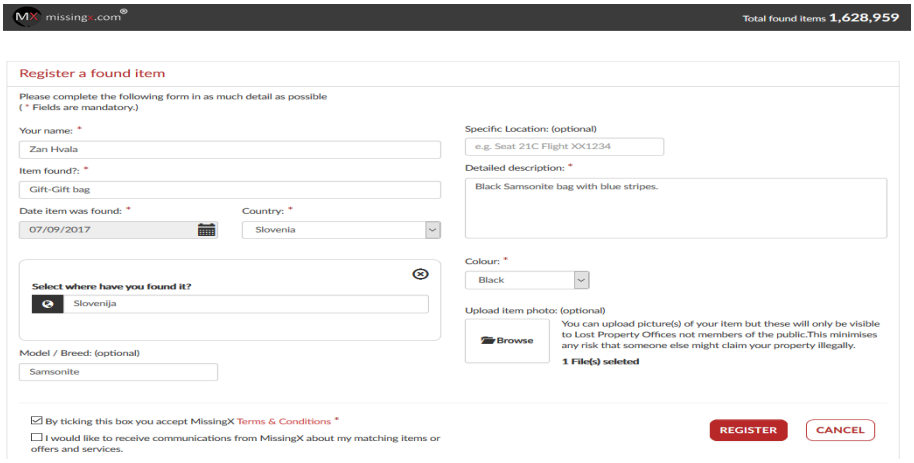
- **Spletna stran LPP** [39] ne omogoča nobene posebne funkcionalnosti. Najdemo lahko le podatek o tem, kam shranjujejo najdene predmete.

- **Spletna stran lostproperty** [38] (slika 2.3) je namenjena iskanju predmetov, ki so se izgubili na letališčih in železniških postajah na območju Velike Britanije. V obrazec vpišemo svoje kontaktne podatke in lastnosti izgubljenega predmeta. S klikom na gumb "Submit" potrdimo vpisane podatke. Izvede se preusmeritev na spletno stran, kjer je prikazan povzetek vpisanih podatkov. Vidimo tudi informacijo, ki nam pove, da nas v primeru najdbe izgubljenega predmeta v njihovi bazi podatkov o tem obvestijo. Oblika, v kateri hranijo podatke o predmetih, uporabnikom ni vidna.



Slika 2.3: Spletna stran lostproperty.

- **Spletna stran missingx [40]** (slika 2.4) je trenutno največja platforma za iskanje najdenih in izgubljenih predmetov. Iskanje poteka na podoben način kot na spletni strani lostproperty. Ko izberemo, če želimo iskati med najdenimi ali izgubljenimi predmeti, se odpre obrazec v katerega vpišemo državo, datum izgube oziroma najdbe in opis predmeta. S klikom na gumb "Next" nas spletna stran vodi na naslednji obrazec, kjer izberemo lokacijo (najdbe oziroma izgube predmeta). Ob kliku na gumb "Search" se izvede preusmeritev na stran z rezultati. Stran z rezultati ponuja tudi možnost ponovitve iskanja s spremenjenimi podatki. Registriran uporabnik lahko prijavi izgubljen ali najden predmet. Prijava predmeta poteka preko obrazca, kjer vpišemo svoje ime, opis predmeta, datum najdbe oziroma izgube, državo, model ali znamko, podrobnejši opis in barvo predmeta. Priložimo lahko tudi sliko. Glede na vpisane podatke se prikažejo dodatne možnosti opisa predmeta (na primer: tip, serijska številka, proizvajalec,...). Predmet prijavimo s klikom na gumb "Register" ali prekličemo prijavo s klikom na gumb "Cancel". Podatki o predmetih so predstavljeni v HTML obliki.



Slika 2.4: Spletna stran missingx.

Poglavje 3

Opis uporabljenih tehnologij

Poglavje opisuje tehnologije uporabljene pri izdelavi spletne aplikacije. Na koncu poglavja je opisana tudi programska oprema s pomočjo katere smo omenjene tehnologije lahko optimalno uporabili.

3.1 Programski jezik Java

Programski jezik Java je eden izmed najbolj popularnih programskih jezikov na svetu [32]. Uporablja se predvsem za razvoj spletnih aplikacij temelječih na konceptu odjemalec-strežnik. Je objektno usmerjen in prenosljiv programski jezik. Razvil ga je James Gosling s sodelavci v podjetju Sun Microsystems. Objektno programiranje temelji na konceptu objektov, ki vsebujejo podatke v obliki atributov in metode, ki s temi podatki manipulirajo [20]. Prenosljivost pomeni, da napisano in prevedeno Java bitno kodo (ang. bytecode) lahko izvajamo na kateremkoli sistemu z JVM (Java Virtual Machine). Poznamo tri glavne različice (ang. editions) Jave:

- **Java ME (Micro Edition)** - zajema področje vgrajenih (ang. embedded) in mobilnih (ang. mobile) naprav.
- **Java SE (Standard edition)** - zajema področje namiznih in strežniških okolij.

- **Java EE (Enterprise Edition)** - uporabljena za izdelavo naše spletne aplikacije in v nadaljevanju tudi podrobneje opisana. Namenjena je razvoju obsežnih spletnih poslovnih aplikacij katere zahtevajo skalabilnost, zanesljivost in varnost [15].

3.2 Zbirka Java EE

Java EE je zbirka specifikacij in tehnologij, ki temeljijo na programskem jeziku Java in razširja Java SE platformo [14]. Razvita je bila za namen izdelave obsežnih, distribuiranih, večnivojskih, skalabilnih, zanesljivih in varnih spletnih aplikacij. Za poslovne aplikacije, Java EE uporablja distribuiran večnivojski aplikacijski model, kjer je logika razdeljena v več komponent glede na funkcijo posamezne komponente.

3.2.1 Večnivojska arhitektura

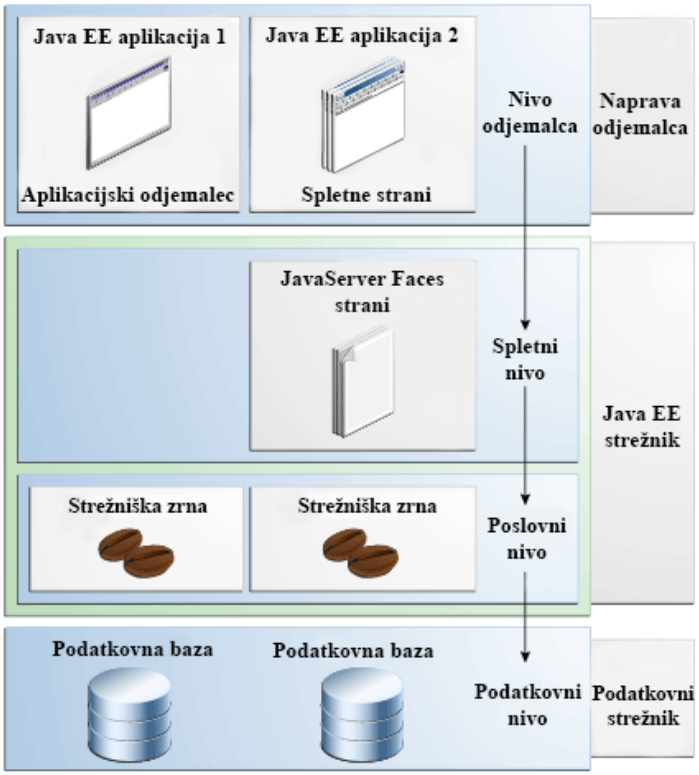
Slika 3.1 prikazuje razdelitev večnivojskih aplikacij v posamezne v nivoje:

- **nivo odjemalca (ang. Client Tier)** - se izvaja na napravi odjemalca.
- **spletni nivo (ang. Web Tier)** - se izvaja na Java EE strežniku.
- **poslovni nivo (ang. Business Tier)** - se izvaja na Java EE strežniku.
- **nivo dostopa do podatkov (ang. EIS - Enterprise Information System Tier)** - se izvaja na podatkovnem strežniku.

Večnivojska arhitektura poskrbi za ločitev interne predstavitve podatkov od predstavitve podatkov, ki so prikazani uporabniku.

3.2.2 Ogrodje JavaServer Faces

JavaServer Faces (JSF) predstavlja ogrodje (ang. framework) za izdelavo spletnih spletnih aplikacij, ki temeljijo na Java tehnologiji [23]. Omogoča dodajanje komponent na spletno stran in povezovanje komponent z objekti



Slika 3.1: Večnivojske Java EE aplikacije.

na strani strežnika. Vsebuje API (ang. Application Programming Interface), ki skrbi za predstavitev komponent in upravljanje z njihovim stanjem, upravljanje z dogodki, validacijo na strani strežnika, pretvorbo podatkov itd. Pri razvoju naše aplikacije smo uporabili odprtokodno knjižnico komponent PrimeFaces, katera omogoča hitrejšo in enostavnejšo izdelavo uporabniškega vmesnika.

3.2.3 Komponenta Enterprise JavaBeans - EJB

EJB je strežniška komponenta, ki enkapsulira poslovno logiko aplikacije [31]. EJB vsebnik (ang. container) zagotavlja izvajalno okolje (ang. run-time environment) za EJB zrna na strani aplikacijskega strežnika. Le ta omogočajo lažji razvoj velikih, distribuiranih aplikacij, saj jim EJB vsebnik zagotavlja

storitve na sistemskem nivoju, kot so: upravljanje s transakcijami, varnost, upravljanje življenskega cikla in komunikacija s podatkovno bazo. To pomeni, da se razvijalec lahko osredotoči na reševanje poslovnih problemov in mu ni potrebno skrbeti za delovanje storitev na sistemskem nivoju, ki se izvajajo v ozadju. Ločimo tri vrste EJB zrn in sicer:

- **Sejna zrna (Session Beans)** se delijo na:
  - sejna zrna s stanjem (ang. Stateful Session Beans) - ohranjajo stanje z odjemalcem v času celotne seje,
  - sejna zrna brez stanja (ang. Stateless Session Beans) - ne ohranjajo stanja z odjemalcem in
  - posamezna sejna zrna (ang. Singleton Session Beans) - katerih instanca je ustvarjena samo enkrat in obstajajo v času celotnega življenskega cikla aplikacije.
- **Sporočilna zrna (ang. Message-Driven Beans)** - omogočajo asinhrono pošiljanje in prejemanje sporočil preko sistema JMS (ang. Java Message Service).
- **Entitetna zrna (ang. Entity Beans)** - omogočajo interakcijo aplikacije s podatkovno bazo.

Razred v Javi ustreza definiciji zrna (ang. JavaBean), če ima javen kontruktor brez argumentov ter za dostop in nastavitev lastnosti oziroma atributov uporablja za to namenjene privatne metode. Razred mora med drugim implementirati vmesnik `Serializable`.

3.3 Podatkovna baza MySQL

MySQL je sistem za upravljanje s podatkovnimi bazami [19]. Je odprtokodna implementacija relacijske podatkovne baze, ki za delo s podatki uporablja strukturiran poizvedovalni jezik SQL (ang. Structured Query Language). Deluje na več različnih platformah kot so: Microsoft Windows [26],

Linux [17], macOS [25], FreeBSD [24] in še mnoge druge. Nekatere izmed lastnosti MySQL sistema so: uporaba več procesorjev hkrati, API za različne programske jezike (C, C++, Java, PHP, Python,...) ter uporaba shranjenih procedur in sprožilcev.

3.4 Aplikacijski strežnik WildFly

Za delovanje naše spletne aplikacije potrebujemo aplikacijski strežnik, ki implementira specifikacije platforme Java EE. Izbrali smo aplikacijski strežnik WildFly napisan v programskem jeziku Java, kateri podpira Java EE 7 platformo. Pomembne funkcije strežnika WildFly so:

- možnost izvajanja v gruči računalniških sistemov (ang. computer cluster),
- podpora ogrodju Hibernate [21] za povezavo med objekti in podatkovno bazo ter
- podpora JAAS (ang. Java Authentication and Authorization Service) za avtentikacijo in avtorizacijo [7].

3.5 HTML, CSS in JSON

HTML (ang. Hypertext Markup Language) je standarden označevalni jezik za izdelavo spletnih strani in aplikacij [27]. Opisuje zgradbo in semantično strukturo spletnega dokumenta. Spletni brskalnik pridobi HTML dokument iz spletnega strežnika ali lokalnega diska in ga nato prikaže v obliki spletne strani. HTML dokument lahko ustvarimo ali urejamo v kateremkoli urejevalniku besedil, zapisan pa je v obliki HTML elementov sestavljenih iz značk.

Predloge, ki določajo izgled spletnih strani (angl. Cascading Style Sheets) so predstavljene v obliki slogovnega jezika in skrbijo za prezentacijo spletne strani [37]. Z njihovo pomočjo definiramo stil HTML elementov in njihov prikaz na spletni strani. Določamo lahko pisave, barve, velikosti, odmike,

poravnave in še mnogo ostalih atributov. Podpirajo tudi nadziranje nad aktivnostmi na primer prekritje elementa z miško.

Format za izmenjavo podatkov JSON se uporablja za prenos podatkovnih objektov med strežnikom in odjemalcem [11]. Podatkovni objekt je predstavljen kot zbirka parov ime-vrednost. Je tekstoven format, neodvisen od programskega jezika. Njegova prednost je enostavnost.

3.6 Programska oprema

Za urejanje programske kode smo uporabljali integrirano razvojno okolje (ang. integrated development environment) IntelliJ IDEA razdeljeno na brezplačno različico IntelliJ IDEA Community Edition in plačljivo različico IntelliJ IDEA Ultimate Edition [13]. Uporabili smo plačljivo različico razvojnega okolja, saj brezplačna različica ne nudi podpore Java EE specifikaciji, ki je ključna pri razvoju naše spletne aplikacije. Plačljiva različica ponuja:

- avtomatsko dopolnjevanje kode,
- navigacijo do izvora s klikom na del kode,
- integracijo s sistemi za nadzor različic kode kot so CVS [34], Git [35], Subversion [36] in ostali,
- direkten dostop do podatkovnih baz ORACLE [29], PostgreSQL [30] in MySQL,
- podporo aplikacijskim strežnikom Geronimo [1], GlassFish [2], JBoss (WildFly), Jetty [3], Tomcat [4], WebLogic [5] in WebSphere [6] in
- podporo različnim programskim jezikom (Java, Go, Perl, XML, HTML, PHP, Python, SQL,...).

Za komunikacijo s podatkovno bazo MySQL smo uporabljali integrirano razvojno okolje za MySQL podatkovne baze, MySQL Workbench. Glavne funkcionalosti orodja so načrtovanje, administracija in spremljanje stanja podatkovne baze na grafičen način.

Poglavje 4

Analiza zahtev, načrtovanje in arhitektura

Pred začetkom razvoja spletne aplikacije smo naredili analizo funkcionalnih in nefunkcionalnih zahtev. S pomočjo rezultatov analize smo načrtovali ER diagram, ki je logična predstavitev podatkovnega modela in služi prikazu entitet, njihovih atributov in povezav med njimi. Dokončan ER diagram smo z generirano SQL skripto preslikali v relacijski podatkovni model in tako ustvarili shemo podatkovne baze. Bolj podroben opis analize zahtev, načrtovanja in arhitekture aplikacije sledi v nadaljevanju poglavja.

4.1 Analiza zahtev

Analize zahtev smo se lotili tako, da smo določili funkcionalne in nefunkcionalne zahteve aplikacije. Naš glavni cilj je, zadostiti zahtevam, da lahko postanemo krovna spletna aplikacija za iskanje predmetov.

4.1.1 Funkcionalne zahteve

V začetni fazi smo se odločili razdeliti potencialne uporabnike sistema na različne vloge: obiskovalec (neregistriran uporabnik), registriran uporabnik

in administrator. Vsaki vlogi je dodeljenih več funkcionalnosti, nekatere pa so skupne vsem. Podrobna razdelitev funkcionalnosti je sledeča:

- **Obiskovalec (neregistriran uporabnik)** :
  - registracija,
  - pregled nad izgubljenimi ali najdenimi predmeti in njihovimi podrobnostmi,
  - sortiranje predmetov po več atributih,
  - iskanje predmetov glede na vpisan iskalni niz in
  - kontaktiranje uporabnika, ki je dodal izgubljen ali najden predmet.
- **Registriran uporabnik** :
  - vse funkcionalnosti obiskovalca,
  - dodajanje novega izgubljenega ali najdenega predmeta,
  - nastavitev dodanega predmeta, da ni več viden obiskovalcem,
  - spreminjanje podatkov dodanega predmeta,
  - izbris dodanega predmeta in
  - urejanje lastnega profila.
- **Administrator** :
  - vse funkcionalnosti registriranega uporabnika,
  - potrditev ustreznosti prikaza predmetov (brez potrditve administratorja dodan predmet ni viden ostalim uporabnikom),
  - urejanje in brisanje predmetov,
  - ogled revizijske sledi predmeta,
  - dodelitev ali odvzem vlog uporabnikom,
  - izbris uporabnikov in spreminjanje njihovih podatkov,

– arhiviranje in

– uvoz predmetov iz arhiva.

4.1.2 Nefunkcionalne zahteve

Pomemben del analize zahtev predstavljajo nefunkcionalne zahteve, ki opisujejo obnašanje aplikacije. Osredotočili smo se na varnost, možnost obnovitve podatkov, dokumentacijo programske kode in grafično podobo.

Varnost

Varnost je ena najpomembnejših nefunkcionalnih zahtev. V primeru naše aplikacije smo varnost razdelili na avtentikacijo in avtorizacijo [8]. Avtentikacija je preverjanje pristnosti uporabnika. Primer avtentikacije je ustrezen vpisu uporabniškega imena in gesla, s katerima se uporabnik prijavi v aplikacijo. Z uporabniškim imenom se identificira, z geslom pa preveri pristnost uporabnika. Avtorizacija se zgodi po avtentikaciji in predstavlja ugotavljanje pravic uporabnika v aplikaciji. Pove nam, do katerih funkcij aplikacije lahko avtenticiran uporabnik dostopa in pod katerimi pogoji.

Za namen avtentikacije in avtorizacje uporabnikov smo uporabili JAAS, ki razširja obstoječo varnostno arhitekturo Jave [9]. V nastavitveno datoteko `standalone.xml` aplikacijskega strežnika smo dodali sledečo programsko kodo:

Nastavitvena datoteka standalone.xml.

```
...
<security-domain name="jBossJaasLostAndFoundMysqlRealm">
  <authentication>
    <login-module
        code="org.jboss.security.auth.spi.DatabaseServerLoginModule"
        flag="required">
      <module-option name="dsJndiName"
          value="java:/LostAndFoundMysql"/>
```

```
    <module-option name="principalsQuery" value="SELECT
       u.password FROM user u where u.username =?"/>
    <module-option name="rolesQuery" value="SELECT r.rolename,
       'Roles' FROM role r INNER JOIN user_has_role uhr ON
       r.roleid = uhr.role_roleid INNER JOIN user u on u.userid
       = uhr.user_userid WHERE u.username =?"/>
    <module-option name="hashAlgorithm" value="SHA-256"/>
    <module-option name="hashEncoding" value="base64"/>
   </login-module>
  </authentication>
</security-domain>
...
```

Elementu varnostne domene (ang. security domain) smo določili ime, po katerem bo domena razpoznavna. Element avtentikacije vsebuje prijavni modul na osnovi podatkovne baze. Prijavnemu modulu smo določili naslednje atribute:

- **dsJndiName** - predstavlja JNDI (ang. Java Naming and Directory Interface) ime podatkovnega vira (ang. data source) iz katerega črpamo podatke o uporabnikih,
- **principalsQuery** - predstavlja poizvedbo SQL za pridobitev gesla uporabnika,
- **rolesQuery** - predstavlja poizvedbo SQL za pridobitev vlog uporabnika,
- **hashAlgorithm** - predstavlja zgoščevalni algoritem za kriptiranje gesel in
- **hashEncoding** - specificira tip kodiranja [18].

Entitete uporabljene v poizvedbah so podrobneje opisane v nadaljevanju poglavja, kjer predstavimo relacijski podatkovni model.

Vsebina nastavitvene datoteke `jboss-web.xml` določa katera varnostna domena bo uporabljena v naši spletni aplikaciji.

Nastavitvena datoteka jboss-web.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<jboss-web>
  <security-domain>
    java:/jaas/jBossJaasLostAndFoundMysqlRealm
  </security-domain>
</jboss-web>
```

Naslednji del programske kode opisuje varnostno omejitev za vlogo administratorja.

Nastavitvena datoteka web.xml.

```
<security-constraint>
  <display-name>Protected admin</display-name>
  <web-resource-collection>
    <web-resource-name>Protected area only for
       admin</web-resource-name>
    <url-pattern>/views/protected_admin/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>admin</role-name>
  </auth-constraint>
  <user-data-constraint>
    <transport-guarantee>NONE</transport-guarantee>
  </user-data-constraint>
</security-constraint>

<security-role>
  <description>Administrator</description>
  <role-name>admin</role-name>
</security-role>

<security-role>
  <description>Registered user</description>
  <role-name>user</role-name>
</security-role>
```

V elementu `security-constraint` definiramo pravice dostopa do strani in podstrani spletne aplikacije. `Web-resource-collection` predstavlja seznam URL vzorcev[1] in HTTP operacij, do katerih lahko preko izbrane vloge dostopamo. Privzeto se upoštevajo vse HTTP metode, kot so GET, PUT, POST in DELETE, lahko pa podamo tudi posamezno metodo. `Auth-constraint` definira ali bo avtentikacija uporabljena in kateri vlogi dodelimo pravice dostopa. `User-data-constraint` opiše kako so podatki zavarovani, ko se prenašajo med odjemalcem in strežnikom. Elementi `security-role` pa predstavljajo varnostne vloge uporabljene v aplikaciji. Iz zgornjega odseka programske kode lahko razberemo, da uporabniki z vlogo 'admin' lahko dostopajo do vseh strani na poti `imeGostitelja:številkaVrat/views/protected_admin/imeStrani`. Uporabnikom, ki nimajo vloge admin, pa je dostop do prej omenjenih strani onemogočen.

Obnovitev podatkov

Za primer nenamerne izgube podatkov smo v aplikacijo vgradili možnost obnovitve izgubljenih podatkov o predmetih. Vsak dan ob polnoči se podatki o predmetih v formatu JSON arhivirajo v za to namenjeno mapo. Administrator lahko arhivirane podatke enostavno obnovi preko uporabniškega vmesnika aplikacije, ima pa tudi možnost, da kadarkoli sproži postopek arhiviranja.

Časovnik za arhiviranje podatkov.

```
/**
* Časovnik je namenjen shranjevanju najdenih, izgubljenih in
    vseh predmetov v JSON datoteke, katere lahko kasneje
* preberemo in iz njih obnovimo podatke.
*/
@Singleton
public class ArchiveItemsToJSONTimer {
```

[1]Del URL naslova, ki sledi imenu gostitelja in številki vrat preko katerih je aplikacija dostopna.

```
  private static final Logger LOG =
      Logger.getLogger(ArchiveItemsToJSONTimer.class.getName());

  @EJB
  private LostAndFoundLocal lostAndFoundLocal;

  @Schedule(minute = "0", hour = "0", dayOfWeek = "*")
  public void archiveAllItemsToJSON() {
    LOG.info("Začetek arhiviranja predmetov v JSON datoteke.");
    lostAndFoundLocal.archiveAllItemsToJSON(); // arhiviranje
        vseh predmetov
    lostAndFoundLocal.archiveAllLostItemsToJSON(); //
        arhiviranje izgubljenih predmetov
    lostAndFoundLocal.archiveAllFoundItemsToJSON(); //
        arhiviranje najdenih predmetov
    LOG.info("Arhiviranje predmetov je zaključeno.");
  }
}
```

Konkretno implementacijo arhiviranja podatkov predstavlja časovnik `ArchiveItemsToJsonTimer`. Naš časovnik je avtomatski, kar pomeni, da se ustvari takoj ob uspešni namestitvi aplikacije na aplikacijski strežnik. Z anotacijo `@Singleton` povemo, da naj se ustvari samo ena instanca razreda, ki je dostopna skozi celoten življenski cikel aplikacije. Anotacija `@Schedule` pa določa čas proženja metode `archiveAllItemsToJSON()`, katere naloga je arhiviranje vseh, izgubljenih in najdenih predmetov v datoteke katerih ime je sestavljeno iz niza: `allItems-/allFoundItems-/allLostItems-_-datum_čas`.

Dokumentacija programske kode

Programski kodi je priporočjivo dodati komentarje z opisom delovanja razreda, metode ali vmesnika, saj je tako koda bolj razumljiva razvijalcem, ki želijo popraviti, dodati ali razširiti funkcionalnosti aplikacije. Uporabili smo JavaDoc Sync dodatek za okolje IntelliJ IDEA [10]. Ta nam olajša do-

kumentiranje programske kode. Za generiranje dokumentacije metode po v naprej določeni predlogi se pomaknemo na ime metode in pritisnemo tipki `Alt+Enter`. V meniju nato izberemo `'Generate JavaDoc based on method signature'`. Spodaj je podan primer predloge za dokumentiranje metode brez parametrov:

Predloga za dokumentiranje metode brez parametrov.

```
/**
* Metoda ${NAME} je namenjena ...
*/
```

Spremenljivka `${NAME}` predstavlja ime metode, ki ji podamo komentar. Preostali del besedila razvijalec dopolni s svojim komentarjem, ki opisuje čemu je metoda namenjena.

Grafična podoba

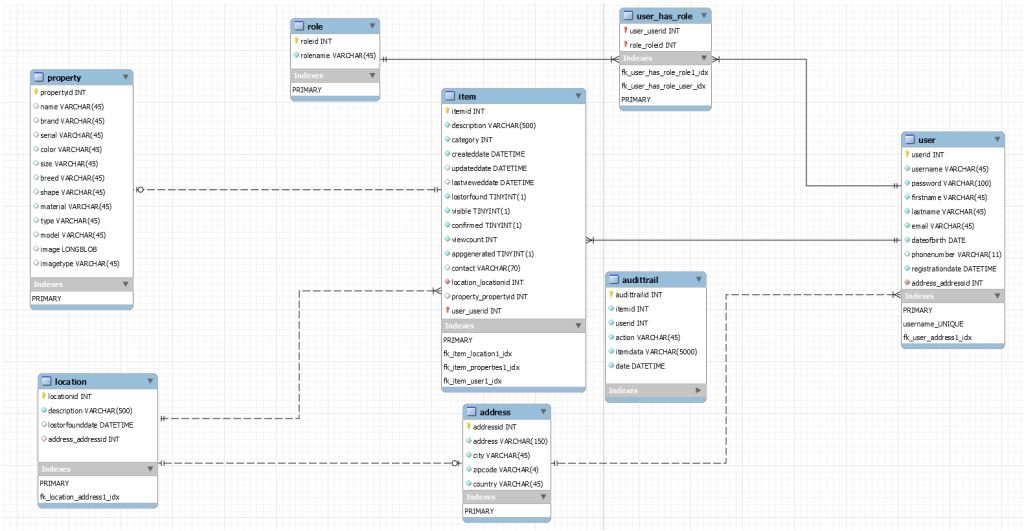
Celoten izgled spletne aplikacije temelji na temi ogrodja PrimeFaces [22]. Izbrali smo si temo bootstrap [43]. Za prikaz komponent (gumbi, tabele, slike, besedila,...) uporabljamo rdečo, modro in belo barvo ter različne odtenke črne barve. Zaradi majhnega števila barv in enotnosti komponent je aplikacija bolj pregledna in prijazna uporabniku. Za kostumizacijo in enoličnost izgleda uporabljamo lasten stil, katerega pravila smo definirali v CSS datoteki.

4.2 Načrt in arhitektura

Načrt bomo prikazali na osnovi relacijskega podatkovnega modela. ER model predstavlja vse entitete in povezave med njimi, ki so potrebne za komunikacijo aplikacije s podatkovno bazo. V sklopu opisa arhitekture podrobneje opišemo vzorec MVC, na katerem temelji delovanje uporabniškega vmesnika naše spletne aplikacije.

4.2.1 Relacijski podatkovni model

Na podlagi preslikave ER diagrama na sliki 4.1 v SQL skripto je ustvarjena shema podatkovne baze. Le ta opredeljuje tabele, njihove atribute oziroma imena stolpcev, podatkovne tipe posameznih stolpcev in integritetne omejitve. To so pogoji, ki jih morajo podatki izpolnjevati, da so lahko zapisani v podatkovno bazo.



Slika 4.1: ER diagram relacijskega podatkovnega modela.

Entitete in njihovi atributi

- **Uporabnik (`user`)** : predstavlja entiteto uporabnika
  - `userid`: avtomatsko generiran enoličen identifikator,
  - `username`: uporabniško ime,
  - `password`: kodirano geslo,
  - `firstname`: ime,
  - `lastname`: priimek,
  - `email`: email naslov,

  - `dateofbirth`: datum rojstva,
  - `phonenumber`: telefonska številka,
  - `registrationdate`: datum registracije in
  - `address_addressid`: identifikator entitete naslova.

- **Vloga (`role`)** : predstavlja entiteto vloge
  - `roleid`: avtomatsko generiran enoličen identifikator in
  - `rolename`: ime vloge.

- **Uporabnik ima vlogo (`user_has_role`)** : predstavlja entiteto, ki povezuje uporabnika z eno ali več vlogami
  - `user_userid`: identifikator entitete uporabnika in
  - `role_roleid`: identifikator entitete vloge.

- **Naslov (`address`)** : predstavlja entiteto naslova (uporabljeno tako pri uporabniku, kot pri lokaciji predmeta)
  - `addressid`: avtomatsko generiran enoličen identifikator,
  - `address`: ulica in hišna številka,
  - `city`: mesto/kraj,
  - `zipcode`: poštna številka in
  - `country`: država.

- **Lokacija (`location`)** : predstavlja entiteto lokacije (predmeta)
  - `locationid`: avtomatsko generiran enoličen identifikator,
  - `description`: opis,
  - `lostorfounddate`: datum izgube/najdbe predmeta in
  - `address_addressid`: identifikator entitete naslova.

- **Lastnost (`property`)** : predstavlja entiteto lastnosti (predmeta)

  - `propertyid`: avtomatsko generiran enoličen identifikator,
  - `name`: ime predmeta,
  - `brand`: znamka predmeta,
  - `serial`: serijska številka predmeta,
  - `color`: barva predmeta,
  - `size`: velikost predmeta,
  - `breed`: pasma predmeta,
  - `shape`: oblika predmeta,
  - `material`: material predmeta,
  - `type`: tip predmeta,
  - `model`: model predmeta,
  - `image`: slika predmeta in
  - `imagetype`: tip slike predmeta.

- **Predmet (`item`)** : predstavlja entiteto izgubljenega ali najdenega predmeta
  - `itemid`: avtomatsko generiran enoličen identifikator,
  - `description`: besedilo opisa,
  - `category`: kategorija, v katero spada predmet (1 - igrača, 2 - nakit, 3 - osebni predmet, 4 - prevozno sredstvo, 5 - tehnologija, 6 - žival, 7 - drugo),
  - `createddate`: datum kreiranja,
  - `lastvieweddate`: datum zadnjega ogleda podrobnosti,
  - `lostorfound`: oznaka pove, ali predmet spada med izgubljene ali najdene predmete (0 - najden, 1 -izgubljen),
  - `visible`: oznaka vidnosti (0 - predmet ni viden, 1 - predmet je viden),

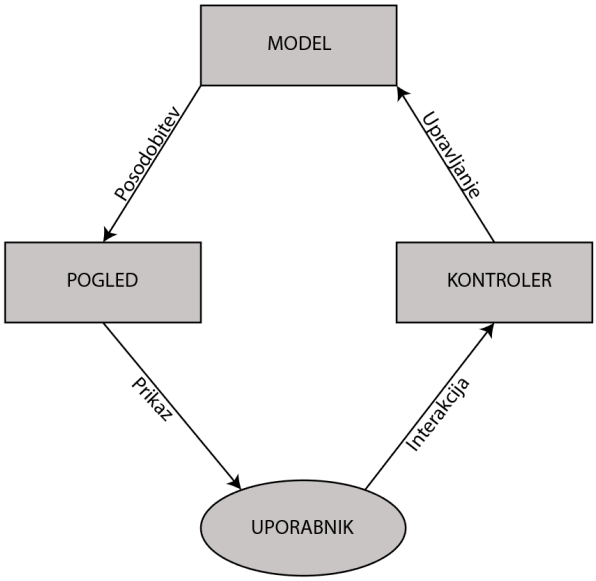
  - `confirmed`: oznaka, ki pove, ali je bil predmet odobren za prikaz s strani administratorja (0 - ni odobren, 1 - je odobren),
  - `appgenerated`: oznaka, ki pove ali je bil predmet dodan preko naše aplikacije ali pridobljen iz drugih virov (0 - pridobljen iz drugih virov, 1 - dodan preko naše aplikacije),
  - `contact`: kontakt uporabnika, ki je dodal predmet,
  - `viewcount`: števec števila ogledov podrobnosti,
  - `location_locationid`: identifikator entitete lokacije,
  - `property_propertyid`: identifikator entitete lastnosti in
  - `user_userid`: identifikator entitete uporabnika, ki je dodal predmet.

- **Revizijska sled (`audittrail`)** : predstavlja entiteto revizijske sledi (predmeta)
  - `audittrailid`: avtomatsko generiran enoličen identifikator,
  - `itemid`: identifikator predmeta,
  - `userid`: identifikator uporabnika,
  - `action`: akcija izvedena nad predmetom (dodajanje, spreminjanje, brisanje, potrjevanje),
  - `itemdata`: podatki o predmetu v tekstovni obliki in
  - `date`: datum zapisa v revizijsko sled.

4.2.2 Arhitektura

Naša spletna aplikacija temelji na arhitekturnem vzorcu MVC. Vzorec deli aplikacijo na tri medsebojno povezane komponente in sicer na model (ang. model), pogled (ang. view) in kontroler (ang. controller), katerih sodelovanje je prikazano na sliki 4.2. Model skrbi za podatke, pogled skrbi za predstavitev podatkov modela uporabniku, kontroler pa izvaja poslovno logiko. Tako

poskrbimo za ločitev interne predstavitve podatkov od predstavitve podatkov prikazanih uporabniku.



Slika 4.2: Diagram sodelovanja komponent po vzorcu MVC.

Poglavje 5

Implementacija

Poglavje opisuje potek izvedbe najzanimivejših funkcionalnosti spletne aplikacije. Začetni del poglavja prikazuje povezavo med objekti v obliki entitetnih zrn in podatki v podatkovni bazi ter medsebojni vpliv med njimi. Nadaljujemo z opisom razvoja prijave, odjave in registracije uporabnika, prikazom revizijske sledi predmeta ter predstavitvijo razvoja funkcionalnosti, ki so specifične za vlogo administratorja. V glavnem delu opišemo najpomembnejši del aplikacije, ki zajema iskanje izgubljenih in najdenih predmetov in s tem povezane funkcionalnosti, kot je na primer pridobivanje podatkov o predmetih iz drugih spletnih strani. Na koncu poglavja sledi še opis postopka testiranja spletne aplikacije in opis težav, ki so nastale med razvojem.

5.1 Komunikacija aplikacije s podatkovno bazo

V nastavitveni datoteki aplikacijskega strežnika `standalone.xml` smo za podatkovni vir določili lokalno MySQL podatkovno bazo, katere relacijski podatkovni model smo opisali v prejšnjem poglavju. Da lahko to podatkovno bazo uporabimo kot podatkovni vir naše aplikacije, smo v datoteko `persistence.xml` dodali sledečo programsko kodo:

Nastavitvena datoteka persistence.xml.

```
<persistence-unit name="LostAndFoundPU">
```

```
  <jta-data-source>java:/LostAndFoundMysql</jta-data-source>
  <class>com.lostandfound.entities.UserEntity</class>
  <class>com.lostandfound.entities.AddressEntity</class>
  <class>com.lostandfound.entities.RoleEntity</class>
  <class>com.lostandfound.entities.LocationEntity</class>
  <class>com.lostandfound.entities.PropertyEntity</class>
  <class>com.lostandfound.entities.ItemEntity</class>
  <class>com.lostandfound.entities.AuditTrailEntity</class>
  <exclude-unlisted-classes>true</exclude-unlisted-classes>
</persistence-unit>
```

Elementu `persistence-unit` smo določili enolično ime, znotraj elementa pa določili lokacijo podatkovnega vira in množico entitetnih zrn upravljanih s strani upravljalca entitetnih zrn (ang. entity manager [44]). Upravljalec entitetnih zrn je aplikacijski programski vmesnik, preko katerega nad tabelami v podatkovni bazi izvajamo poizvedbe, dodajamo nove zapise, brišemo obstoječe zapise ali pa posodobimo vrednosti že obstoječih zapisov. Entitetno zrno predstavlja tabelo v podatkovni bazi, instanca entitetnega zrna pa posamezen zapis oziroma vrstico v tabeli. Primer entitetnega zrna za tabelo uporabnik (`user`):

Primer entitetnega zrna.

```
@Entity
@Table(name = "user")
@NamedQueries({
  @NamedQuery(name = UserEntity.FIND_BY_USERNAME, query =
      "SELECT c FROM UserEntity c WHERE c.username = :username")
})
public class UserEntity implements Serializable {

  private static final long serialVersionUID =
      -3471297528112849474L;

  public static final String FIND_BY_USERNAME =
      "UserEntity.findByUsername";
```

```
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "userid")
    private Integer userId;

    @Column(name = "username")
    private String username;

    @Column(name = "dateofbirth")
    private LocalDate dateOfBirth;

    @ManyToOne(optional = false)
    @JoinColumn(name="address_addressid")
    private AddressEntity address;

    // preostali atributi tabele ter getterji in setterji
}
```

Z anotacijo `@Entity` povemo, da razred predstavlja entitetno zrno. Sledi anotacija `@Table`, kateri določimo ime tabele s katero želimo povezati našo entiteto in anotacija `@NamedQueries` v kateri definiramo eno ali več statičnih poizvedb v jeziku JPQL (ang. Java Persistence Query Language [16]). Sledijo anotacije, ki predstavljajo lastnosti stolpcev v tabeli:

- `@Id` - označuje primarni ključ entitete,
- `@GeneratedValue` - označuje strategijo generiranja vrednosti primarnega ključa, v našem primeru je to IDENTITY, kar pomeni avtomatsko povečevanje vrednosti,
- `@Column` - označuje povezavo spremenljivke s stolpcem v tabeli,
- `@ManyToOne` predstavlja povezavo mnogo proti ena (med tabelama `user` in `address`) in
- `@JoinColumn` - označuje tuji ključ tabele [28].

Na enak način, kot smo kreirali entitetno zrno za tabelo `user`, smo to naredili za vse ostale tabele v podatkovni bazi. Tako smo lahko nadaljevali z razvojem aplikacije.

5.2 Prijava, odjava in registracija uporabnika

V fazi razvoja smo implementirali tudi možnost prijave, odjave in registracije uporabnika. V meniju v zgornjem delu spletne aplikacije se nahaja gumb "Prijava". Ob kliku na gumb se odpre prijavni dialog (prijavno okno) prikazan na sliki 5.1 z dvema vnosnima poljema, v katera uporabnik vpiše svoje uporabniško ime in geslo.

Slika 5.1: Prijavni dialog.

Spodnji primer prikazuje sestavo pogleda (ang. view) prijavnega dialoga implementiranega s pomočjo ogrodja JSF in knjižnice PrimeFaces.

Primer pogleda prijavnega dialoga.

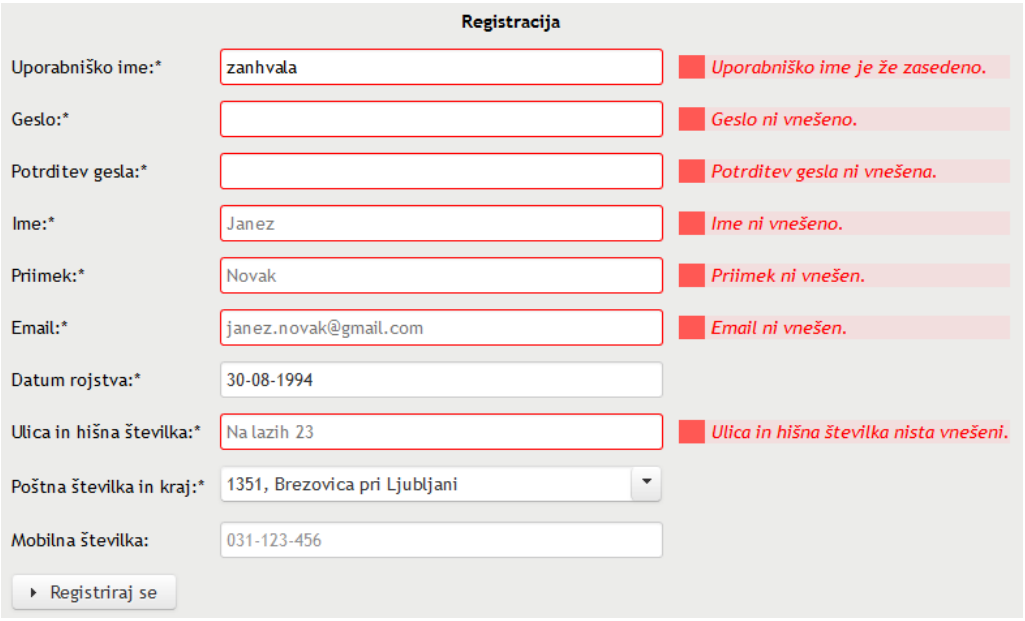
```
<h:form id="formLogInDialog">
  <p:dialog header="#{msg['dialog.header.log.in']}"
     widgetVar="logInDlg" dynamic="true" resizable="false"
     closeOnEscape="true" modal="true">
  <p:messages id="messagesId" />
  <p:ajax event="close" update=":formLogInDialog" />

  <p:panelGrid id="userData" columns="2" cellpadding="5"
     styleClass="ui-noborder pPanelGridDefault">
```

```
    <h:outputLabel value="Uporabniško ime:" for="usernameId" />
    <p:inputText id="usernameId" value="#{userBean.username}"
        size="45"
      required="true" requiredMessage="Uporabniško ime ni
          vnešeno." label="Uporabniško ime">
      <f:validateLength minimum="0" maximum="45" />
    </p:inputText>

    <h:outputLabel value="Geslo:" for="passwordId" />
    <p:password id="passwordId" value="#{userBean.password}"
        size="45"
      required="true" requiredMessage="Geslo ni vnešeno."
          label="Geslo">
      <f:validateLength minimum="0" maximum="100" />
    </p:password>
  </p:panelGrid>

  <p:commandButton value="Prijava" icon="ui-icon-person"
    actionListener="#{userBean.logIn}"
    oncomplete="if (!args.validationFailed &amp;&amp;
        args.loggedIn) PF('logInDlg').hide();"
    update=":headerForm messagesId userData" />
  <p:commandButton value="Zapri" icon="ui-icon-closethick"
      onclick="PF('logInDlg').hide()" />
  </p:dialog>
</h:form>
```

Pogled vsebuje komponente v HTML obliki primerne za prikaz v spletnem brskalniku. Ob kliku na gumb "Prijava" se sproži postopek prijave uporabnika v spletno aplikacijo. V prvi fazi preverimo, da polji nista prazni in njuna vsebina ni predolga - v nasprotnem primeru javimo ustrezno sporočilo. Po ustrezni validaciji polj se sproži poslušalec akcije (ang. action listener) definiran v 21. vrstici, ki pokliče metodo `logIn()`, ki je implementirana v zrnu `UserBean`. Zrno je označeno z anotacijo `@SessionScoped`, kar pomeni, da je dosegljivo celoten čas HTTP seje. Če med izvajanjem metode `logIn()` ne pride do napak, zapremo dialog in uporabnik je tako prijavljen v aplika-

cijo. V primeru vpisa napačnega uporabniškega imena ali gesla uporabniku napako javimo preko ustreznega sporočila, ki se prikaže na vrhu prijavnega dialoga. Po uspešni prijavi se uporabniku v meniju ponudijo možnosti urejanja profila in odjave iz aplikacije.

Za uspešno prijavo je prej seveda potrebna registracija. Obrazec za registracijo je prikazan na sliki 5.2. S klikom na gumb "Registriraj se" se



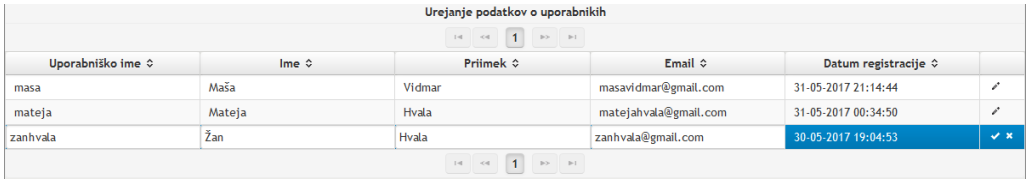
Slika 5.2: Obrazec za registracijo uporabnika.

sproži postopek registracije, medtem pa se v ozadju izvajajo že prej omenjena preverjanja in klici metod, preko katerih podatke uporabnika zapišemo v podatkovno bazo. Ob uspešni registraciji novo registriranega uporabnika avtomatsko prijavimo v aplikacijo in ga preusmerimo na začetno stran.

5.3 Naloge administratorja

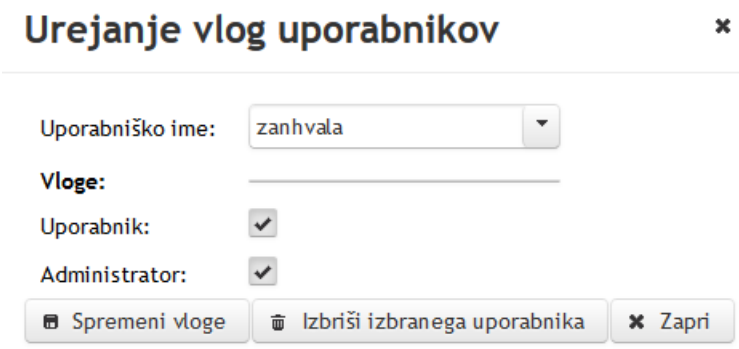
Zgoraj opisan postopek registracije uporabniku avtomatsko dodeli vlogo "uporabnik". Naša aplikacija podpira tudi vlogo "administrator", kateri so omogočene dodatne funkcionalnosti. Ena izmed njih je urejanje podatkov o uporab-

nikih. Administratorju je dostopna posebna stran, ki je prikazana na sliki 5.3, kjer so v obliki tabele vidni vsi podatki o registriranih uporabnikih. Podatke o posameznem uporabniku lahko administrator ureja s klikom na ikono svinčnika. Spremembo podatkov uveljavi s klikom na kljukico ali pa razveljavi s klikom na križec.



Slika 5.3: Urejanje podatkov o uporabnikih.

Na isti strani lahko administrator odpre dialog (slika 5.4), preko katerega izbranemu uporabniku dodeli ali odvzame vlogo. Prav tako pa ima možnost izbranega uporabnika izbrisati ter mu tako onemogočiti prijavo v aplikacijo in uporabo funkcionalnosti vezanih na vlogo "uporabnik".



Slika 5.4: Urejanje vlog uporabnikov.

Naloga administratorja je tudi odobritev na novo dodanih predmetov. Ko nekdo doda nov predmet v seznam izgubljenih ali najdenih predmetov, je potrebno počakati, da administrator predmet odobri. To zagotavlja, da vsebina podatkov o predmetu ni sporna. Šele po odobritvi je predmet viden vsem uporabnikom spletne aplikacije.

5.4 Iskanje predmetov

Glavna naloga naše spletne aplikacije je omogočanje enostavnega iskanja izgubljenih in najdenih predmetov. Ob vstopu na začetno stran se obiskovalcu poleg možnosti prijave in registracije ponudi tudi možnost obiska strani z izgubljenimi ali najdenimi predmeti. V predelu noge spletne strani je prikazana trenutna verzija aplikacije. Izgled začetne strani je prikazan na sliki 5.5.

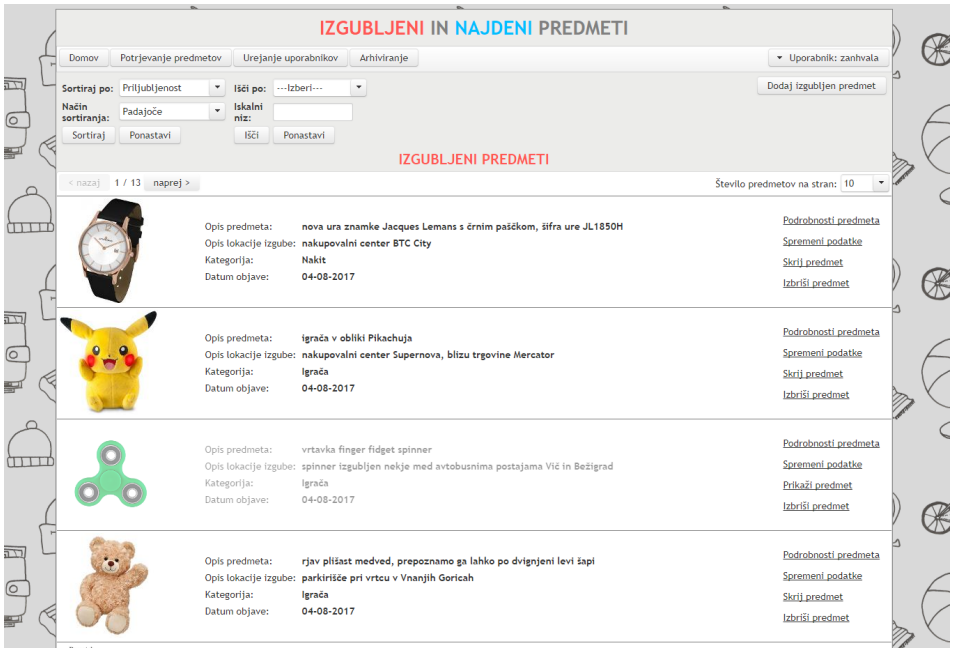
Slika 5.5: Začetna stran.

5.4.1 Pregled predmetov

Ob obisku strani z izgubljenimi predmeti se obiskovalcu prikaže seznam izgubljenih predmetov. Predmet je predstavljen s svojimi lastnostmi in sliko na katero lahko kliknemo, da se prikaže večja različica slike. Določene lastnosti (opis predmeta, opis lokacije izgube, točen naslov izgube, poštna številka in kraj izgube, datum izgube, slika predmeta ter kategorija) so skupne vsem predmetom, ostale pa so odvisne od kategorije. Predmet lahko spada v sledeče kategorije:

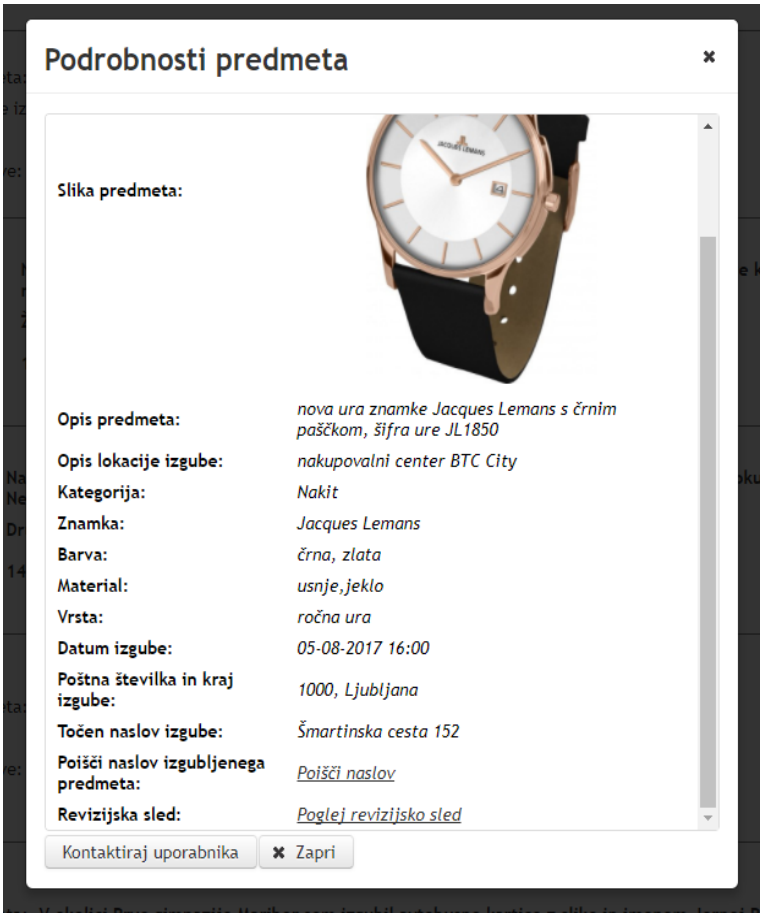
- **igrača** z lastnostmi: znamka, barva, velikost, oblika in material,
- **nakit** z lastnostmi: znamka, barva, material in vrsta,
- **osebni predmet** z lastnostmi enakimi kot nakit,
- **prevozno sredstvo** z lastnostmi: znamka, serijska številka, barva, vrsta in model,
- **tehnologija** z lastnostmi enakimi kot prevozno sredstvo,
- **žival** z lastnostmi: ime, barva in pasma ter
- **drugo** z lastnostmi enakimi kot nakit.

Zgoraj predstavljene kategorije smo si izbrali na podlagi pregleda že obstoječih spletnih aplikacij za iskanje predmetov in lastne presoje, katere kategorije so smiselne za uporabo. Naštete kategorije poskušajo zajeti čim večji del različnih predmetov, ki imajo skupne lastnosti.



Slika 5.6: Izgled strani z izgubljenimi predmeti.

Kot lahko razberemo iz slike 5.6 so vsakemu predmetu dodane različne možnosti v obliki povezav. Obiskovalec strani (neregistriran uporabnik) ima na voljo samo povezavo "Podrobnosti predmeta". S klikom na to povezavo se odpre okno prikazano na sliki 5.7, kjer so prikazane tudi lastosti predmeta, ki v seznamu niso vidne. Poleg ogleda vseh lastnosti predmeta lahko s klikom na povezavo "Poišči naslov" v pojavnem oknu s pomočjo Google Maps [12] poiščemo točen naslov predmeta. Obiskovalec lahko sam določi,



Slika 5.7: Pregled podrobnosti predmeta.

koliko predmetov želi videti na eni strani (10, 20 ali 30). Prehod med stranmi seznama predmetov je omogočen s klikom na gumb "nazaj" ali "naprej". Logika prikazovanja predmetov in spreminjanje števila prikazanih predmetov na

posamezni strani je implementirana v razredu `ItemsRepeatPaginator`. Razred hrani seznam vseh izgubljenih predmetov in seznam izgubljenih predmetov, ki so prikazani na eni strani. Glede na izbrano število predmetov na stran se s pomočjo metode `updateModel()` posodablja seznam predmetov, ki je prikazan na posamezni strani.

5.4.2 Revizijska sled

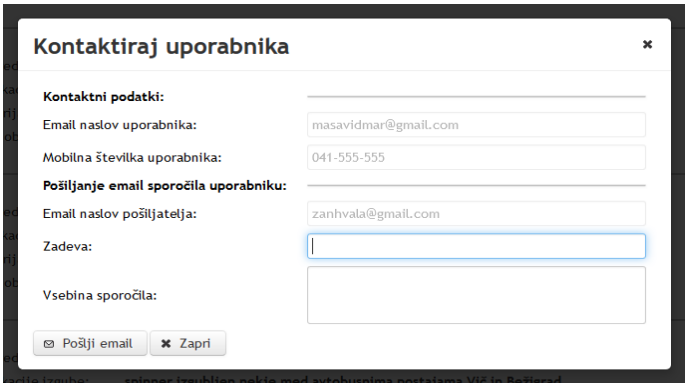
Administrator si lahko ogleda revizijsko sled posameznega predmeta v obliki PDF dokumenta. V revizijsko sled se ob vsaki akciji izvedeni nad predmetom (dodajanje, spreminjanje, potrjevanje ali brisanje) beležijo podatki o predmetu in uporabniku, ki je izvedel akcijo, naziv akcije ter čas izvedbe akcije. Primer revizijske sledi je prikazan na sliki 5.8.

Uporabnik	Akcija	Podatki	Datum
zanhvala	dodajanje	Opis: nova ura znamke Jacques Lemans s črnim paščkom, šifra ure JL1850 Kategorija: Nakit Datum kreiranja: 2017-08-16 Izgubljen/najden: izgubljen Viden: Da Potrjen: Ne Število ogledov: 0 Dodan preko aplikacije: Da Opis lokacije: nakupovalni center BTC City Datum izgube/najdbe: 2017-08-05 Naslov: Ulica in hišna številka: Šmartinska cesta 152 Mesto: Ljubljana Poštna številka: 1000 Država: Slovenija Znamka: Jacques Lemans Barva: črna, zlata Material: usnje,jeklo Tip: ročna ura	2017-08-16 18:21:03
zanhvala	potrditev	Opis: nova ura znamke Jacques Lemans s črnim paščkom, šifra ure JL1850 Kategorija: Nakit Datum kreiranja: 2017-08-16 Izgubljen/najden: izgubljen Viden: Da Potrjen: Da Število ogledov: 0 Dodan preko aplikacije: Da Opis lokacije: nakupovalni center BTC City Datum izgube/najdbe: 2017-08-05 Naslov: Ulica in hišna številka: Šmartinska cesta 152 Mesto: Ljubljana Poštna številka: 1000 Država: Slovenija Znamka: Jacques Lemans Barva: črna, zlata Material: usnje,jeklo Tip: ročna ura	2017-08-16 18:21:07

Slika 5.8: Revizijska sled predmeta.

5.4.3 Kontaktiranje uporabnika

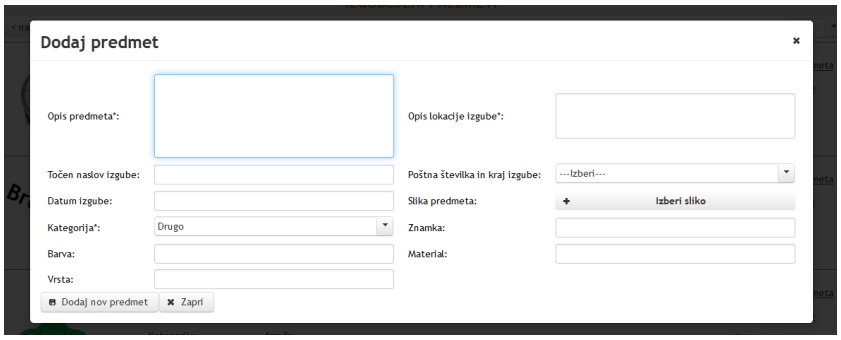
Na voljo je tudi opcija za kontaktiranje uporabnika (slika 5.9), kjer lahko uporabniku, ki je izbran izgubljeni predmet dodal, pošljemo elektronsko sporočilo ali pa ga kontaktiramo preko mobilne številke (mobilna številka se prikaže samo v primeru, če jo je uporabnik ob registraciji vnesel). V našem primeru je elektronski naslov pošiljatelja že določen, saj smo prijavljeni z uporabnikom zanhvala.



Slika 5.9: Možnost kontaktiranja uporabnika.

5.4.4 Dodatne možnosti prijavljenega uporabnika

Prijavljen uporabnik lahko poleg pregleda podrobnosti predmeta tudi spreminja podatke o predmetu, skrije predmet (da ni viden ostalim uporabnikom) ali pa dodane predmete izbriše. Naštete možnosti seveda veljajo le za predmete, ki jih je uporabnik dodal sam. Uporabnik ima tudi možnost dodajanja novih predmetov preko dialoga, ki je prikazan na sliki 5.10. Dialog se odpre s klikom na gumb "Dodaj izgubljen predmet". Ko je predmet uspešno dodan, uporabniku sporočimo, da mora predmet počakati na odobritev s strani administratorja, da bo postal viden v seznamu.



Slika 5.10: Dialog za dodajanje novega izgubljenega predmeta.

5.4.5 Urejanje in iskanje

Seznam prikazanih predmetov lahko spreminjamo z opcijama za urejanje in iskanje. Pri urejanju predmetov določimo lastnost, po kateri želimo sortirati seznam predmetov, in način sortiranja, ki predstavlja vrstni red prikaza (padajoče ali naraščajoče). Spodnji odsek programske kode iz metode za urejanje predmetov prikazuje urejanje po atributu "priljubljenost".

Urejanje predmetov po priljubljenosti.

```
List<Item> listToSort = lost ? new
    ArrayList<>(lostItemList.getOrigModel()) : new
    ArrayList<>(foundItemList.getOrigModel());
switch (selectedSortAttribute) {
  case PRILJUBLJENOST:
    if (NARASCAJOCE.equals(selectedSortMode)) {
      listToSort.sort(Comparator.comparing(Item::getViewCount)
      );
    } else if (PADAJOCE.equals(selectedSortMode)) {
      listToSort.sort(Comparator.comparing(Item::getViewCount)
      .reversed());
    }
    break;
// sortiranje po ostalih atributih

if (lost) {
```

```
  lostItemList.setOrigModel(listToSort);
  lostItemList.updateModel();
} else {
  foundItemList.setOrigModel(listToSort);
  foundItemList.updateModel();
}
```

V primeru, ko želimo poiskati točno določene predmete, uporabimo funkcijo za iskanje po seznamu predmetov. Izbrati moramo lastnosti za iskanje in v za to namenjeno polje vpisati iskalni niz. S klikom na gumb "Išči" se izvede iskanje in posodobi seznam prikazanih predmetov, kateri sedaj vsebuje samo predmete glede na vnešene kriterije za iskanje.

5.4.6 Časovnik za izbris predmetov

Po določenem času predmeti postanejo neaktualni in ni več potrebe, da bi podatke o njih hranili v podatkovni bazi. V ta namen smo implementirali časovnik, katerega metoda za brisanje predmetov je prikazana spodaj.

Metoda časovnika za brisanje predmetov.

```
@Schedule(minute = "0", hour = "0", dayOfWeek = "*")
public void deleteNonVisitedItems() {
  LOG.info("Začetek brisanja predmetov, ki niso bili ogledani
     več kot 30 dni.");
  LocalDateTime dateTimeNow = LocalDateTime.now();
  for (Item item : lostAndFoundLocal.getAllItems()) {
    // če predmet še ni bil ogledan, upoštevamo datum, kdaj je
       bil dodan, drugače upoštevamo datum, kdaj je bil
       nazadnje ogledan
    Duration dur = item.getLastViewedDate() == null ?
    Duration.between(item.getCreatedDate(), dateTimeNow) :
       Duration.between(item.getLastViewedDate(), dateTimeNow);
    // če predmet ni bil ogledan že 30 dni in več ali pa v
       30-tih dneh od kar je bil dodan sploh še ni bil ogledan,
       ga izbrišemo
    if (dur.toDays() > 30) {
```

```
            lostAndFoundLocal.deleteItem(item.getItemId());
        }
    }
    LOG.info("Brisanje predmetov je zaključeno.");
}
```

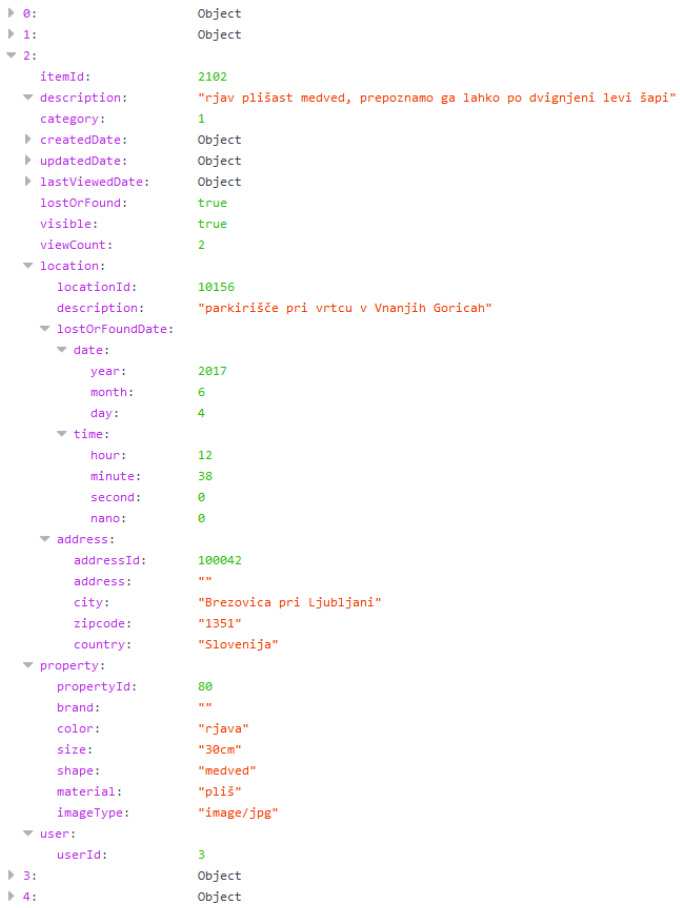
Zapomnimo si trenuten datum in ga primerjamo z datumom zadnjega ogleda predmeta ali pa z datumom, ki nam pove, kdaj je bil predmet dodan. Datum za primerjavo izberemo tako, da pogledamo ali je bil predmet sploh že ogledan, in upoštevamo ta datum, sicer pa upoštevamo datum dodajanja predmeta. Za vsak predmet se vprašamo, če je med datumoma, ki jih primerjamo, preteklo 30 dni ali več in če to drži, podatke o predmetu izbrišemo iz podatkovne baze. Izbrisane podatke o predmetih lahko kadarkoli obnovimo, saj se arhiviranje izvaja vsak dan.

5.4.7 Integracija z zunanjimi sistemi

Naša aplikacija omogoča integracijo z zunanjimi sistemi preko protokola REST (ang. REpresentational State Transfer [33]). REST za svoje delovanje izkorišča standardne metode protokola HTTP (kot so GET, PUT, POST in DELETE). V naši aplikaciji smo s pomočjo HTTP metod GET in PUT ter protokola REST implementirali pridobivanje podatkov o predmetih, dodajanje novih predmetov in posodabljanje podatkov že obstoječih predmetov. Na sliki 5.11 je prikazan del podatkov v formatu JSON, ki so pridobljeni preko URL naslova: `imeGostitelja:vrata/LostAndFound-war/rest/-items/all`.

5.5 Pridobivanje podatkov o predmetih

Ambicija postati centralna spletna stran za iskanje izgubljenih in najdenih predmetov v Sloveniji zahteva redno pridobivanje podatkov o predmetih iz drugih spletnih strani. Za namen prikaza takega delovanja smo si za primer izbrali spletno stran slovenske policije in spletno stran Radio City. Že



Slika 5.11: Podatki o predmetih v formatu JSON.

v 2. poglavju smo na kratko opisali, v kakšni obliki ti dve spletni strani predstavljata podatke o predmetih. Spletna stran slovenske policije ponuja prikaz podatkov o predmetih v HTML obliki, prav tako pa lahko te podatke pridobimo tudi v JSON formatu. Spletna stran Radio City ponuja prikaz in predstavitev podatkov o predmetih samo v HTML obliki. Implementacije smo se lotili z izdelavo dveh časovnikov, ki vsak dan osvežita podatke o predmetih iz prej omenjenih spletnih strani.

Odsek programske kode na naslednji strani predstavlja implementacijo metode časovnika za pridobivanje podatkov o najdenih predmetih s strani slovenske policije.

Metoda časovnika za predmete slovenske policije.

```
@Schedule
public void addPolicijaItems() {
  List<PolicijaItem> allPolicijaItems;
  List<Item> itemsToAdd = new ArrayList<>();

  Client client = ClientBuilder.newClient();
  WebTarget target =
      client.target("http://www.policija.si/apps/predmeti/
                    rest.php/najdeni");
  JsonArray response =
      target.request(MediaType.APPLICATION_JSON).get(JsonArray.class);
  allPolicijaItems = gson.fromJson(response.toString(), new
      TypeToken<List<PolicijaItem>>(){}.getType());

  for (PolicijaItem policijaItem : allPolicijaItems) {
    Item itemToAdd = mapPolicijaItemToItem(policijaItem);
    if (Duration.between(itemToAdd.getCreatedDate(),
        LocalDateTime.now()).toDays() <= 30 &&
        !itemAlreadyInDatabase(itemToAdd)) {
      itemsToAdd.add(itemToAdd);
    }
  }
  for (Item item : itemsToAdd) {
    lostAndFoundLocal.addItem(item, item.getUser());
  }
}
```

Kot lahko razberemo iz programske kode smo na spletno stran poslali zahtevo, katere odgovor predstavlja seznam vseh najdenih predmetov v JSON formatu. Vse pridobljene predmete pretvorimo v objekte tipa `PolicijaItem`. Medtem, ko se sprehajamo čez seznam pretvorjenih predmetov, predmet ponovno pretvorimo v objekt tipa `Item`, ki predstavlja splošen predmet, ter preverimo, da predmet ni starejši od 30 dni in še ni zapisan v podatkovni bazi. Predmete, ki uspešno prestanejo preverjanje, dodamo v seznam in jih zapišemo v podatkovno bazo. To pomeni, da so sedaj vidni v naši spletni

aplikaciji.

Spodnja programska koda predstavlja implementacijo metode časovnika za pridobivanje podatkov o najdenih in izgubljenih predmetih s spletne strani Radio City.

Metoda časovnika za predmete Radio City.

```
@Schedule
public void addRadioCityLostAndFoundItems() throws IOException {
  List<String> lostItemDate = new ArrayList<>();
  List<String> lostItemTitle = new ArrayList<>();
  List<String> lostItemCategory = new ArrayList<>();
  List<String> lostItemContact = new ArrayList<>();
  List<String> lostItemDescription = new ArrayList<>();

  List<String> foundItemDate = new ArrayList<>();
  List<String> foundItemTitle = new ArrayList<>();
  List<String> foundItemCategory = new ArrayList<>();
  List<String> foundItemContact = new ArrayList<>();
  List<String> foundItemDescription = new ArrayList<>();

  fillItemList("http://www.radiocity.si/izgubljeno.php?list=",
     lostItemDate, lostItemTitle, lostItemCategory,
     lostItemContact, lostItemDescription);
  fillItemList("http://www.radiocity.si/najdeno.php?list=",
     foundItemDate, foundItemTitle, foundItemCategory,
     foundItemContact, foundItemDescription);

  List<Item> lostItemsToAdd =
     mapRadioCityItemToItem(lostItemDate, lostItemTitle,
     lostItemCategory, lostItemContact, lostItemDescription,
     true);
  List<Item> foundItemsToAdd =
     mapRadioCityItemToItem(foundItemDate, foundItemTitle,
     foundItemCategory, foundItemContact, foundItemDescription,
     false);

  for (Item item : lostItemsToAdd) {
```

```
      lostAndFoundLocal.addItem(item, item.getUser());
    }

    for (Item item : foundItemsToAdd) {
      lostAndFoundLocal.addItem(item, item.getUser());
    }
}
```

Zaradi predstavitve predmetov izključno v HTML obliki, smo se lotili pridobivanja podatkov na drugačen način kot v primeru slovenske policije. Na začetku smo ustvarili prazne sezname, ki so namenjeni hranjenju lastnosti najdenih in izgubljenih predmetov. S pomočjo metode `fillItemList` prej omenjene sezname napolnimo. To naredimo tako, da metodi kot argument podamo izhodiščni naslov ter se sprehodimo čez vse podstrani tega naslova in sproti polnimo sezname s podatki. Napolnjene sezname pošljemo metodi `mapRadioCityItemToItem`, ki tvori objekte tipa `Item` iz podatkov zapisanih v seznamih. V prej omenjeni metodi preverimo tudi, da predmet ni starejši od 30 dni in še ni zapisan v podatkovni bazi. Kreirane in preverjene predmete (objekte tipa `Item`) na koncu dodamo v podatkovno bazo.

5.6 Testiranje

Testiranje smo izvajali sproti, tako da smo z njim pričeli že na začetku razvoja spletne aplikacije. Večinoma smo testiranje izvajali sami, saj prototip spletne aplikacije teče na lokalnem aplikacijskem strežniku in med razvojem zunaj našega lokalnega omrežja ni bila dostopna. Za vsako novo dodano funkcionalnost smo preverili, če izpolnjuje naša pričakovanja. Preverjanje smo izvajali tako, da smo ob spremembi programske kode spletno aplikacijo naložili na strežnik in preizkusili delovanje novih funkcionalnosti s pomočjo razhroščevalnika in vizualnega prikaza podatkov v podatkovni bazi.

Proti koncu razvoja smo omogočili nekaj poskusnim uporabnikom dostop do naše spletne aplikacije. Prosili smo jih naj aplikacijo preizkusijo in podajo mnenje o njeni uporabnosti. Analiza odzivov poskusnih uporabnikov je

podrobneje opisana v 6. poglavju.

5.7 Težave pri razvoju

Pri razvoju spletne aplikacije smo naleteli na nekaj zanimivih težav.

- Prva težava je bila konfiguracija aplikacijskega strežnika WildFly. Za pravilno delovanje smo morali nastaviti ustrezen podatkovni vir. V našem primeru je to podatkovna baza MySQL. Ustvariti je bilo potrebno uporabnika s pravico dostopa do grafične konzole, preko katere lahko upravlja s celotim aplikacijskim strežnikom.
- Drug izziv je predstavljala pravilna konfiguracija varnosti v obliki avtentikacije in avtorizacije uporabnikov, podrobneje opisana v 4. poglavju.
- Ostale težave, ki so sledile pa so bile večinoma povezane z napakami v programski kodi, katere smo uspešno odpravili s pomočjo razhroščevalnika, ki je vgrajen v naše razvojno okolje.

Poglavje 6

Analiza

Izdelan prototip spletne aplikacije implementira vse funkcionalnosti podane v zahtevah. Namen imamo postati krovna spletna aplikacija za hitro in enostavno iskanje predmetov. Ključne prednosti pred ostalimi že obstoječimi aplikacijami so:

- enoten način pridobivanja podatkov o predmetih iz drugih spletnih strani,
- kreiranje revizijske sledi predmeta za učinkovit nadzor nad izvedenimi akcijami,
- avtomatsko arhiviranje podatkov o predmetih,
- možnost obnovitve izbrisanih podatkov in
- varnost, ki temelji na principu vlog.

Med analizo je bila odkrita tudi pomanjkljivost. Primarno je naša aplikacija namenjena uporabnikom prenosnih in stacionarnih računalnikov. Zaradi velikega zaslona, ki ga imajo omenjene naprave, je aplikacija pregledna in enostavna za uporabo. Ugotovili smo, da bi z izboljšanjem prilagodljivosti mobilnim napravam lahko pridobili večje število uporabnikov.

Priložnost vidimo tudi v oglaševanju aplikacije. S primernim oglaševanjem bi zajeli širšo množico potencialnih uporabnikov in tako naredili aplikacijo bolj prepoznavno.

Analizirali smo možnost objave prototipa aplikacije na spletu. Na začetku bi se aplikacija (tako kot sedaj), izvajala na našem aplikacijskem strežniku. Zakupili bi spletno domeno. Ta bi predstavljala enoličen naslov preko katerega bi uporabniki lahko dostopali do naše spletne aplikacije. Poskrbeli bi tudi za vpis v več spletnih iskalnikov kot so: Google, Najdi.si, Yahoo in Bing, saj bi s tem pridobili na prepoznavnosti. Ko bi videli, da število uporabnikov narašča, bi aplikacijo preselili na oddaljen strežnik. S tem bi zagotovili neprekinjeno delovanje aplikacije (ne bi bili več odvisni od naše spletne povezave in električnega omrežja) in uporabnikom omogočili boljšo uporabniško izkušnjo.

V zadnjem delu analize bi izpostavili nekaj pozitivnih mnenj in nekaj pripomb pridobljenih s strani testnih uporabnikov. Prototip aplikacije je testiralo 5 uporabnikov starih med 20 in 50 let. Za testiranje so imeli na voljo 3 dni.

- **Pozitivna mnenja** :
  - intuitivna in enostavna uporaba,
  - pregleden uporabniški vmesnik in
  - prikaz podatkov o predmetih iz drugih spletnih strani.
- **Pripombe** :
  - občasno počasnejše delovanje in
  - prilagoditev mobilnim napravam.

Vsa našteta mnenja so nam bila in nam še bodo v pomoč pri nadaljnem razvoju in izpopolnjevanju naše spletne aplikacije.

Poglavje 7

Zaključek

V diplomskem delu smo predstavili spletno aplikacijo za iskanje izgubljenih in najdenih predmetov. Trenutno na spletu še ni podobne aplikacije v slovenskem jeziku, kar nam je dalo še dodatno motivacijo pri razvoju tovrstne aplikacije. Poleg iskanja predmetov, ki je glavna naloga naše aplikacije, so dodane tudi funkcionalnosti, ki pripomorejo k boljši uporabniški izkušnji in splošni uporabnosti spletne aplikacije. Med najpomembnejše naloge aplikacije štejemo:

- omogočanje prijave, odjave in registracije uporabnika,
- dodajanje, spreminjanje in izbris predmetov,
- sortiranje in iskanje po seznamu predmetov,
- ogled revizijske sledi predmeta,
- arhiviranje podatkov o predmetih,
- urejanje uporabniškega profila,
- pridobivanje podatkov o predmetih iz drugih spletnih strani in
- nadzor nad uporabniki omogočen samo administratorju.

Izpostavili bi predvsem arhiviranje in pridobivanje podatkov o predmetih iz drugih spletnih strani. Ti dve funkcionalnosti sta glavna prednost pred že obstoječimi aplikacijami za iskanje predmetov. To nam ponuja priložnost postati krovna spletna aplikacija za iskanje predmetov v Sloveniji.

Ustrezno smo poskrbeli tudi za varnost. Ta temelji na principu avtentikacije in avtorizacije. Avtentikacija je predstavljena v obliki vnosa uporabniškega imena in gesla. Preko principa avtorizacije pa uporabniku dodelimo pravice, s katerimi ima možnost dostopa do različnih funkcionalnosti v aplikaciji.

Spletna aplikacija tako ponuja celovito rešitev problema iskanja izgubljenih in najdenih predmetov, seveda pa so možne tudi dodatne nadgradnje oziroma izboljšave.

- Hitrost aplikacije bi lahko izboljšali z optimiziranim načinom branja podatkov iz podatkovne baze, kar bi omogočilo hitrejši prikaz seznama predmetov uporabniku.
- Možnost nadgradnje vidimo na področju avtomatskega prepoznavanja kategorije predmeta glede na njegov opis in sliko. To nam bi omogočilo avtomatiziran nadzor nad pravilnostjo podatkov o predmetih.
- Zaradi razširjenosti mobilnih naprav predvidevamo optimizacijo prikaza spletne aplikacije na mobilnih napravah.

Literatura

[1] Aplikacijski strežnik Geronimo. Dosegljivo: http://geronimo.apache.org/. [Dostopano: 29. 8. 2017].

[2] Aplikacijski strežnik GlassFish. Dosegljivo: http://www.oracle.com/technetwork/middleware/glassfish/overview/index.html. [Dostopano: 29. 8. 2017].

[3] Aplikacijski strežnik Jetty. Dosegljivo: http://www.eclipse.org/jetty/. [Dostopano: 29. 8. 2017].

[4] Aplikacijski strežnik Tomcat. Dosegljivo: https://tomcat.apache.org/. [Dostopano: 29. 8. 2017].

[5] Aplikacijski strežnik WebLogic. Dosegljivo: https://www.oracle.com/middleware/weblogic/index.html. [Dostopano: 29. 8. 2017].

[6] Aplikacijski strežnik WebSphere. Dosegljivo: http://www-03.ibm.com/software/products/en/appserv-was. [Dostopano: 29. 8. 2017].

[7] Aplikacijski strežnik WildFly. Dosegljivo: http://wildfly.org/about/. [Dostopano: 29. 7. 2017].

[8] Avtentikacija in avtorizacija. Dosegljivo: https://sl.wikipedia.org/wiki/Informacijska_varnost. [Dostopano: 29. 8. 2017].

[9] Danny Coward. *Java EE 7: The Big Picture*. McGraw-Hill Education, 2015.

[10] Dodatek JavaDoc Sync. Dosegljivo: `https://plugins.jetbrains.com/plugin/6340-javadoc-sync-plugin-10`. [Dostopano: 29. 8. 2017].

[11] Format datotek JSON. Dosegljivo: `http://www.json.org/json-sl.html`. [Dostopano: 29. 7. 2017].

[12] Google Maps. Dosegljivo: `https://maps.google.com/`. [Dostopano: 29. 8. 2017].

[13] Integrirano razvojno okolje IntelliJ IDEA. Dosegljivo: `https://en.wikipedia.org/wiki/IntelliJ_IDEA`. [Dostopano: 29. 7. 2017].

[14] Java EE. Dosegljivo: `http://www.oracle.com/technetwork/java/javaee/overview/index.html`. [Dostopano: 29. 7. 2017].

[15] Java EE Tutorial. Dosegljivo: `https://docs.oracle.com/javaee/7/tutorial/`. [Dostopano: 29. 7. 2017].

[16] Jezik JPQL. Dosegljivo: `http://docs.oracle.com/html/E13946_04/ejb3_langref.html`. [Dostopano: 29. 8. 2017].

[17] Operacijski sistem Linux. Dosegljivo: `https://www.linux.org/`. [Dostopano: 29. 7. 2017].

[18] Francesco Marchioni Michal Cmil, Michal Matloka. *Java EE 7 Development with WildFly*. Packt Publishing, 2014.

[19] MySQL. Dosegljivo: `https://sl.wikipedia.org/wiki/MySQL`. [Dostopano: 29. 7. 2017].

[20] Objektno programiranje. Dosegljivo: `https://en.wikipedia.org/wiki/Object-oriented_programming`. [Dostopano: 29. 7. 2017].

[21] Ogrodje Hibernate. Dosegljivo: http://hibernate.org/. [Dostopano: 29. 7. 2017].

[22] Ogrodje PrimeFaces. Dosegljivo: https://www.primefaces.org/. [Dostopano: 29. 8. 2017].

[23] Ogrodje JavaServer Faces. Dosegljivo: http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html. [Dostopano: 29. 7. 2017].

[24] Operacijski sistem FreeBSD. Dosegljivo: https://www.freebsd.org/. [Dostopano: 29. 7. 2017].

[25] Operacijski sistem macOS. Dosegljivo: https://en.wikipedia.org/wiki/MacOS. [Dostopano: 29. 7. 2017].

[26] Operacijski sistem Microsoft Windows. Dosegljivo: https://www.microsoft.com/sl-si/windows/. [Dostopano: 29. 7. 2017].

[27] Označevalni jezik HTML. Dosegljivo: https://en.wikipedia.org/wiki/HTML. [Dostopano: 29. 7. 2017].

[28] Peter A. Pilgrim. *Java EE 7 Developer Handbook*. Packt Publishing, 2013.

[29] Podatkovna baza ORACLE. Dosegljivo: https://www.oracle.com/database/index.html. [Dostopano: 29. 8. 2017].

[30] Podatkovna baza PostgreSQL. Dosegljivo: https://www.postgresql.org/. [Dostopano: 29. 8. 2017].

[31] Poslovna Java zrna - Enterprise JavaBeans. Dosegljivo: https://en.wikipedia.org/wiki/Enterprise_JavaBeans. [Dostopano: 29. 7. 2017].

[32] Programski jezik Java. Dosegljivo: https://sl.wikipedia.org/wiki/Programski_jezik_javai. [Dostopano: 29. 8. 2017].

[33] Protokol REST. Dosegljivo: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm. [Dostopano: 29. 8. 2017].

[34] Sistem za nadzor različic CVS. Dosegljivo: http://www.nongnu.org/cvs/. [Dostopano: 29. 8. 2017].

[35] Sistem za nadzor različic git. Dosegljivo: https://git-scm.com/. [Dostopano: 29. 8. 2017].

[36] Sistem za nadzor različic Subversion. Dosegljivo: https://subversion.apache.org/. [Dostopano: 29. 8. 2017].

[37] Slogovni jezik CSS. Dosegljivo: https://sl.wikipedia.org/wiki/CSS. [Dostopano: 29. 7. 2017].

[38] Spletna stran lostproperty. Dosegljivo: https://www.lostproperty.org/search.php. [Dostopano: 7. 9. 2017].

[39] Spletna stran LPP. Dosegljivo: http://www.lpp.si/javni-prevoz/najdeni-predmeti. [Dostopano: 29. 7. 2017].

[40] Spletna stran missingx. Dosegljivo: https://www.missingx.com/. [Dostopano: 7. 9. 2017].

[41] Spletna stran policije. Dosegljivo: http://www.policija.si/apps/predmeti/. [Dostopano: 29. 7. 2017].

[42] Spletna stran Radio City. Dosegljivo: http://www.radiocity.si/izgubljeno.php. [Dostopano: 29. 7. 2017].

[43] Tema Bootstrap. Dosegljivo: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework). [Dostopano: 29. 8. 2017].

[44] Upravljalec entitetnih zrn. Dosegljivo: https://docs.oracle.com/javaee/7/api/javax/persistence/EntityManager.html. [Dostopano: 29. 8. 2017].